AIAA'85

AIAA-85-0360

RECENT IMPROVEMENTS IN EFFICIENCY, ACCURACY, AND CONVERGENCE FOR IMPLICIT APPROXIMATE FACTORIZATION ALGORITHMS

Thomas H. Pulliam
and Joseph L. Steger
NASA Ames Research Center, Moffett Field, CA

AIAA 23rd Aerospace Sciences Meeting

January 14-17, 1985/Reno, Nevada

Recent Improvements in Efficiency, Accuracy, and Convergence for Implicit Approximate Factorization Algorithms

THOMAS H. PULLIAM[1]

AND

JOSEPH L. STEGER[2]

NASA AMES RESEARCH CENTER
MOFFETT FIELD, CALIFORNIA

I) INTRODUCTION

General purpose centrally space differenced implicit finite difference codes in two[1] and three[2] dimensions have been developed at NASA Ames and have been widely distributed since their introduction in 1977 and 1978. These codes, now referred to ARC2D and ARC3D, can run either in inviscid or viscous mode for steady or unsteady flow. They use general coordinate systems and can be run on any smoothly varying curvilinear mesh, even a mesh that is quite skew. Because they use well ordered finite difference grids, the codes can take advantage of vectorized computer processors and have been implemented for the Control Data 205 and the CRAY 1-S and X-MP. On a single processor of the X-MP a vectorized version of the code runs approximately 20 times faster than the original code which was written for the Control Data 7600.

Traditionally gains in computational efficiency due to improved numerical algorithms have kept pace with gains due to increased computer power. Since the ARC2D and ARC3D codes were introduced, a variety of algorithmic changes have been individually tested and have been shown to improve overall computational efficiency. These include use of a spatially varying time step (Δt), use of a sequence of mesh refinements to establish approximate solutions, implementation of various ways to reduce inversion work, improved numerical dissipation terms, and more implicit treatment of terms. Although the various individual algorithm improvements can interact with each other, sometimes adversely making optimization difficult, their combined effect has lead to an order of magnitude gain in computational efficiency for steady state applications. This is a gain equivalent to that achieved with computer hardware. Unsteady flow calculations have also benefited from some of the above improvements.

The purpose of this paper is to describe the above algorithm improvements and to access and quantify (as much as possible) their advantages and disadvantages. The improvements that we consider are constrained to be relatively simple so that they do not unduly complicate our general purpose flow solvers.

The main theme of this paper is to demonstrate that by using a series of established and simple procedures, we can maintain a single straightforward general purpose computer code that is competitive with specialized codes. For example, the steady state convergence obtained with this code for inviscid Euler equations is competitive with codes that use multigrid, and much better than current viscous multigrid codes. Likewise the same code can be used for unsteady flow simulation, and only codes using similar algorithms are competitive with it for unsteady viscous flow simulation.

[1] Research Scientist

[2] Senior Staff Scientist, Associate Fellow AIAA

II) BACKGROUND

General purpose implicit, approximately factored, finite difference codes have been generated for solving the Euler or thin layer Navier-Stokes equations in general coordinates. If ξ, η, and ς denote the curvilinear independent variables, then these equations can be written as

$$\partial_t \widehat{Q} + \partial_\xi \widehat{E} + \partial_\eta \widehat{F} + \partial_\varsigma \widehat{G} = Re^{-1} \partial_\varsigma \widehat{S} \tag{1}$$

where

$$\widehat{Q} = J^{-1} \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ e \end{bmatrix}, \quad \widehat{E} = J^{-1} \begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ U(e+p) - \xi_t p \end{bmatrix},$$

$$\widehat{F} = J^{-1} \begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ V(e+p) - \eta_t p \end{bmatrix}, \quad \widehat{G} = J^{-1} \begin{bmatrix} \rho W \\ \rho u W + \varsigma_x p \\ \rho v W + \varsigma_y p \\ \rho w W + \varsigma_z p \\ W(e+p) - \varsigma_t p \end{bmatrix}$$

with

$$\begin{aligned} U &= \xi_t + \xi_x u + \xi_y v + \xi_z w \\ V &= \eta_t + \eta_x u + \eta_y v + \eta_z w \\ W &= \varsigma_t + \varsigma_x u + \varsigma_y v + \varsigma_z w \end{aligned}$$

and

$$\widehat{S} = J^{-1} \begin{bmatrix} 0 \\ \mu m_1 u_\varsigma + (\mu/3) m_2 \varsigma_x \\ \mu m_1 v_\varsigma + (\mu/3) m_2 \varsigma_y \\ \mu m_1 w_\varsigma + (\mu/3) m_2 \varsigma_z \\ \mu m_1 m_3 + (\mu/3) m_2 (\varsigma_x u + \varsigma_y v + \varsigma_z w) \end{bmatrix}$$

Here $m_1 = \varsigma_x^2 + \varsigma_y^2 + \varsigma_z^2$, $m_2 = \varsigma_x u_\varsigma + \varsigma_y v_\varsigma + \varsigma_z w_\varsigma$, and $m_3 = (u^2 + v^2 + w^2)_\varsigma / 2 + Pr^{-1} (\gamma - 1)^{-1} (a^2)_\varsigma$.

Pressure is related to the conservative flow variables, Q, by the equation of state

$$p = (\gamma - 1) \left(e - \frac{1}{2} \rho (u^2 + v^2 + w^2) \right)$$

The metric terms are defined as

$$\begin{aligned} \xi_x &= J(y_\eta z_\varsigma - y_\varsigma z_\eta), & \eta_x &= J(z_\xi y_\varsigma - y_\xi z_\varsigma) \\ \xi_y &= J(z_\eta x_\varsigma - z_\varsigma x_\eta), & \eta_y &= J(x_\xi z_\varsigma - z_\xi x_\varsigma) \\ \xi_z &= J(x_\eta y_\varsigma - y_\eta x_\varsigma), & \eta_z &= J(y_\xi x_\varsigma - x_\xi y_\varsigma) \\ \varsigma_x &= J(y_\xi z_\eta - z_\xi y_\eta), & \xi_t &= -x_\tau \xi_x - y_\tau \xi_y - z_\tau \xi_z \\ \varsigma_y &= J(z_\xi x_\eta - x_\xi z_\eta), & \eta_t &= -x_\tau \eta_x - y_\tau \eta_y - z_\tau \eta_z \\ \varsigma_z &= J(x_\xi y_\eta - y_\xi x_\eta), & \varsigma_t &= -x_\tau \varsigma_x - y_\tau \varsigma_y - z_\tau \varsigma_z \end{aligned}$$

with

$$J^{-1} = x_\xi y_\eta z_\zeta + x_\zeta y_\xi z_\eta + x_\eta y_\zeta z_\xi - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi$$

A noniterative approximate factorization implicit method adapted from Beam and Warming[3] or Briley and McDonald[4] is given for Eq.(1) by

$$\left[I + h\delta_\xi \widehat{A}^n - hD_i|_\xi\right]\left[I + h\delta_\eta \widehat{B}^n - hD_i|_\eta\right]$$
$$\left[I + h\delta_\zeta \widehat{C}^n - hRe^{-1}\overline{\delta_\zeta} J^{-1}\widehat{M}^n J - hD_i|_\zeta\right]\Delta\widehat{Q}^n = \tag{2}$$
$$-h\left(\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n + \delta_\zeta \widehat{G}^n - Re^{-1}\overline{\delta_\zeta}\widehat{S}^n + D_e\widehat{Q}^n\right)$$

where $h = \Delta t$ and A, B, C, and M are the Jacobian matrices, $\frac{\partial E}{\partial Q}, \frac{\partial F}{\partial Q}, \frac{\partial G}{\partial Q}$, and $\frac{\partial S}{\partial Q}$. These Jacobian matrices are given in the Appendix. Note that M, contains derivatives in η Here δ_ξ, δ_η, and δ_ζ are typically three point central difference approximations that are second-order accurate, so the factored left hand side operators form block tridiagonal matrices. The operators ∇ and Δ are simple backward and forward differences. For the viscous terms, a midpoint operator $\overline{\delta_\zeta}$ is used to maintain a compact three point second-order accurate central difference scheme. Fourth-order accurate central Pade difference operators[3] that do not break down the left hand side tridiagonal structure have also been used. Often fourth-order space accuracy is obtained by only altering the RHS difference operators[2], and in this regard, pseudo-spectral operators[5] have been used as well. In the original code development[1,2] numerical dissipation terms denoted as D_i and D_e were added as

$$D_e = \epsilon_e \Delta t J^{-1}[(\nabla\Delta)^2_\xi + (\nabla\Delta)^2_\eta + (\nabla\Delta)^2_\zeta]J \tag{3a}$$

and

$$D_i|_\xi = \epsilon_i \Delta t J^{-1}(\nabla\Delta)_\xi J, \quad D_i|_\eta = \epsilon_i \Delta t J^{-1}(\nabla\Delta)_\eta J, \quad D_i|_\zeta = \epsilon_i \Delta t J^{-1}(\nabla\Delta)_\zeta J \tag{3b}$$

The operators D_i which are inserted into the respective implicit block operators, are central three point second-differences. The implicit numerical dissipation operators are included to stabilize the explicitly treated fourth-difference terms while at the same time keep the LHS factors block tridiagonal. As the D_i work on ΔQ, accuracy is not impaired.

The above equations have been presented in three-dimensional form and a two-dimensional form is easily obtained as a subset.

In the basic ARC2D and ARC3D codes the boundary conditions were treated explicitly. This gives us a simpler more flexible code since all the boundary conditions are isolated from the implicit inversions. The user is responsible for the implementation of boundary conditions. In a later section we will discuss some specific boundary condition implementation and in particular boundary conditions employed for airfoil calculations.

The above algorithm has been widely applied, and, as noted in the introduction, its structure lends itself to vectorized coding. Generally vectorized coding is obtained using an approach where strings of block tridiagonal matrices are inverted simultaneously.

Because of the structure of the algorithm given by Eq.(2), the resultant computer codes are very modular and therefore easily changed. Programming for computer vectorization tends to complicate these codes, but users have generally considered the computer codes to be relatively straightforward.

III) ALGORITHM CHANGES

A series of changes have been implemented into the algorithm to improve steady state efficiency and reduce inversion work. The algorithm changes are constrained so that they do not unduly complicate the basic computer code - even if some loss of efficiency occurs. As a result, the same code can be used for inviscid or viscous and steady or unsteady flow by using simple switches.

The changes implemented into the basic code which improve efficiency, stability and convergence include:

a) space varying Δt

b) use of a mesh refinement sequence to establish an approximate solution

c) reduction of the block tridiagonal matrix inversion work by using simpler matrices that maintain stability through similarity transformation

d) combinations of dissipation operators that improve robustness without impairing accuracy

e) more implicit treatment of dissipation terms.

We are also concerned with improving the overall accuracy of the computational codes, and to this end we have implemented

a) better boundary condition procedures, such as characteristic conditions

b) improved artificial dissipation operators which eliminate spatial oscillations, especially near shocks.

Higher order accurate difference formulas in space, such as fourth-order accurate space operators and perturbation forms[6] which improve global accuracy and eliminate first-order errors have been implemented, but are not maintained in the basic codes.

To help quantify these improvements we shall often contrast their effects on the computation of inviscid flow around a NACA0012 airfoil at a free stream Mach number, $M_\infty = 0.8$ and an angle of attack, $\alpha = 1.25°$. Figure. 1 shows the two-dimensional transonic airfoil computation using ARC2D with all the current improvements. The grid is an 'O' mesh with 192 points around the airfoil and 33 points in the near radial direction. Clustering is used at the leading and trailing edges and in the vicinity of the expected shocks on the upper and lower surface, see in Fig. 1a. The converged coefficient of pressure on the surface is shown in Fig. 1b, Mach contours in Fig. 1c, and pressure contours are shown in Fig. 1d. The convergence history showing the l_2 norm of the residual (RHS of Eq. (2)) plotted against fine grid iteration count is given in Fig. 1e, and rapid convergence for lift is demonstrated in Fig. 1f.

A) Space Varying Δt

If only a steady state solution is required, one can let h (or Δt) change in space. This approach can be viewed as a way to condition the iteration matrix of the relaxation scheme defined via Eq.(2). Use of a space varying Δt can also be interpreted as an attempt to use a more uniform Courant number throughout the field. In any event, changing Δt can be effective for grid spacings that vary from very fine to very coarse - a situation usually encountered in aerodynamic simulations where grids contain a wide variety of length scales.

A space varying Δt has been used in both explicit and implicit schemes (e.g. Shang and Hankey[7], McDonald and Briley[8], Shirnivasan et al[9], Coakley[10], Jameson[11], etc). As a rule one wishes to adjust Δt at each point proportional to the grid spacing and the characteristic speed of the flow. Something like the Courant number restriction (which for the Euler equations in multi-dimensions is a bit of an approximation).

For highly stretched grids the space variation of the grid is the most important parameter to scale with. In subsonic and transonic flow the characteristic speeds have moderate variation and we have found that a purely geometric variation of Δt is adequate, specifically

$$\Delta t = \frac{\Delta t|_{ref}}{1. + \sqrt{J}} \tag{4.1}$$

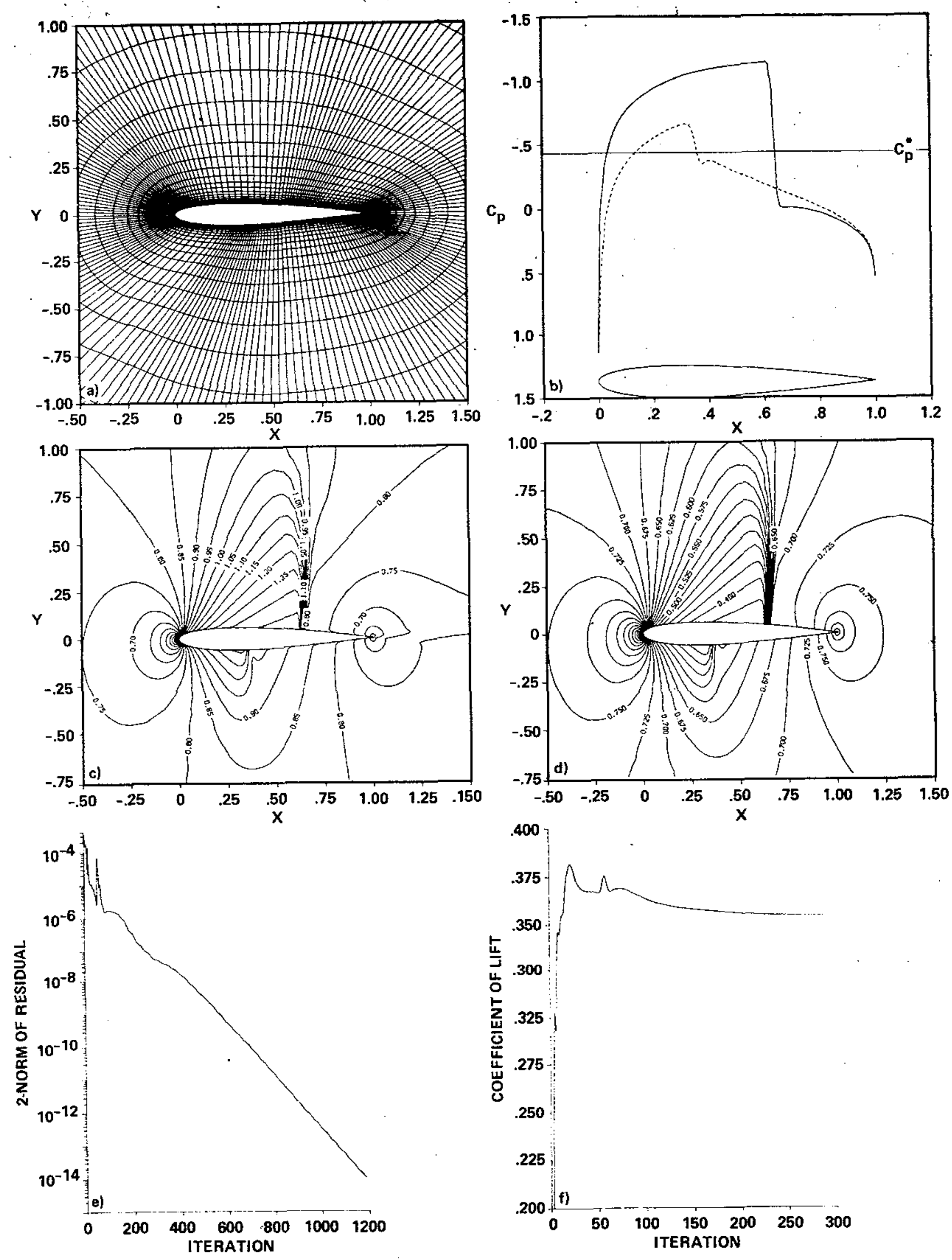


Figure 1. Inviscid NACA 0012 Airfoil Test Case, $M_\infty = 0.8$ Showing Grid, Solution and Convergence History

To illustrate the advantage of using a variable time step, Fig. 2 shows the degradation in convergence rate when a constant step size is substituted for the variable time step in the NACA 0012 test case. For this comparison all other parameters were held constant and no other changes were employed. We should note at this time that the above variation of time step has often worked poorly in viscous flow until the numerical dissipation terms were also put in more implicitly as described later. Also other forms of the variable step size sometimes perform better than Eq. (4.1), for example

$$\Delta t = \frac{\Delta t|_{ref}}{|U| + |V| + a\sqrt{\xi_x^2 + \xi_y^2 + \eta_x^2 + \eta_y^2}} \tag{4.2}$$

which is approximately a constant CFL condition. However, Eq. (4.2) is more costly to compute then Eq. (4.1).

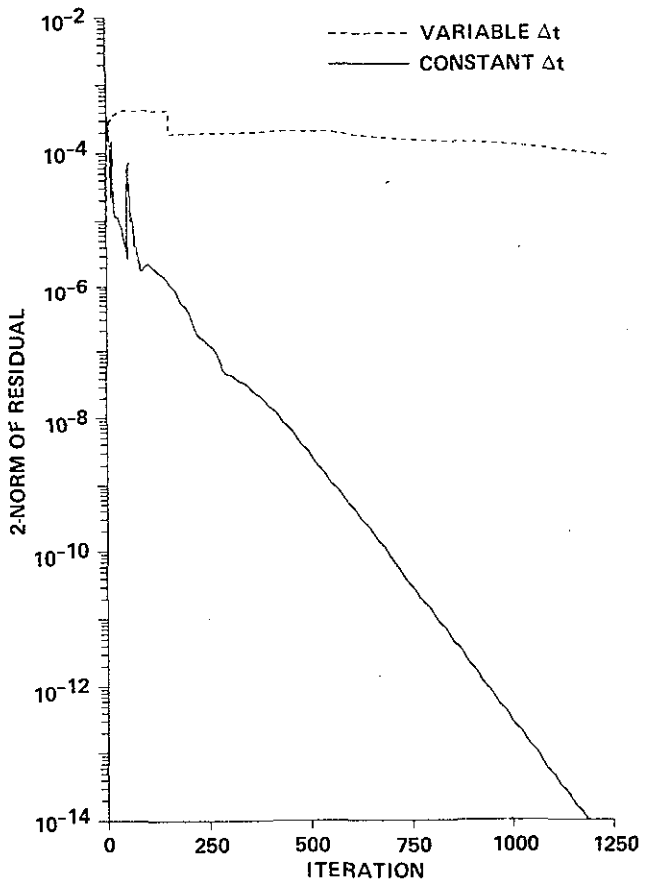


Figure 2. Convergence Improvement Due to Variable Time Step

B) Mesh Sequences

For inviscid airfoil calculations on a grid of O(250 x 50) practical convergence is usually obtained in 500-600 fine grid iterations when the flow field has been started from an initial condition of uniform free stream flow. Typically, the first 100 to 200 iterations on the fine mesh are needed to get past the initial transients which can be a substantial portion of the total solution time. For instance, in the above test case it takes on the order of 600 fine grid iterations for a tight convergence criteria (e.g. lift to 5 decimal places) , 200 of which are spent on clearing out the impulsive start. One common way to accelerate convergence to a steady state is to obtain a good initial guess on the fine mesh by first solving or iterating on a sequence of coarsen grids and then iterpolating the solution up to the next refined grid. Such a mesh sequence procedure can often reduce the amount of time required to obtain a solution to plotable accuracy by a factor of two. Also, because a coarse grid tends to damp high frequency waves, using a mesh sequence procedure can improve the overall robustness of the code.

A mesh sequencing procedure has been implemented in an optionally called stand alone routine. If a sequence of m grids are used, a coarsened grid is cut from each previous grid by halving the number of points in the ξ-direction and by regenerating a new η-distribution of points in the η-direction using a fewer number of points. The η-distribution is regenerated because in viscous flows the spacing near the wall would be too coarsened if the halving procedure is used. A finite number of iterations (perhaps 50) are carried out on each coarsened grid at which point the approximate solution is interpolated onto a more refined grid. The finest grid is then iterated to convergence. The result is faster convergence to practical levels and a more robust starting procedure.

For the NACA 0012 test case a sequence of 3 grids has been used; 48 by 18 and 96 by 25 and the final grid of 192 by 33 points. The convergence of C_l is shown in Fig. 3 to indicate the overall improvement in convergence due to using mesh sequencing in comparison to using a fine grid only. Both cases were started with a free stream initial condition.

c) Simplified Inversion Work

The obvious measure of the computational expense of a algorithm is the total operation count performed on a working code. The operation count of the implicit approximately factored algorithm given by Eq.(2) for inviscid two-dimensional flow is about 1200 operations per point. Of this, approximately 60 % is required for the left hand side factors. Each block 4×4 tridiagonal matrix requires 370 operations per point. For viscous flow the percentage of time required for inversions (i.e. solution time) is less because this cost remains fixed while calculation of viscous terms, the viscous Jacobians and turbulence model terms increase the overall operation count. For 3-D flow simulation a block tridiagonal solution requires 695 operations per point, and as there are three such factors, this means 2085 operations are needed per grid point. For inviscid flow, this is about 70 - 80 % of the total count.

For steady state applications, or when first-order time accuracy is adequate (many forced oscillation cases can be run first-order accurate in time), we can replace the left hand side operators with simplified matrices that maintain the stability properties of the original matrices. Two approaches, both of which rely on similarity transforms, are available: diagonalization[12], and block reduction[13]. Both approaches give the correct steady state solution but are only first-order in time with the diagonalization method being nonconservative in time as well. Only the results using diagonalization are described here, see Ref. 13 for more details on the block reduction scheme.

Starting with the two-dimensional form of Eq. (2) (without the dissipation or viscous terms) we have

$$\left[I + h\delta_\xi \widehat{A}^n\right]\left[I + h\delta_\eta \widehat{B}^n\right] \Delta\widehat{Q}^n = -h\left(\delta_\xi \widehat{E}^n + \delta_\eta \widehat{F}^n\right) = \widehat{R}^n \tag{5.1}$$

The flux Jacobians $\widehat{A}$ and $\widehat{B}$ each have real eigenvalues and a complete set of eigenvectors. Therefore, the Jacobian matrices can be diagonalized, see Warming, Beam and Hyett[14],

$$\Lambda_\xi = T_\xi^{-1}\widehat{A}T_\xi \quad \text{and} \quad \Lambda_\eta = T_\eta^{-1}\widehat{B}T_\eta \tag{5.2}$$

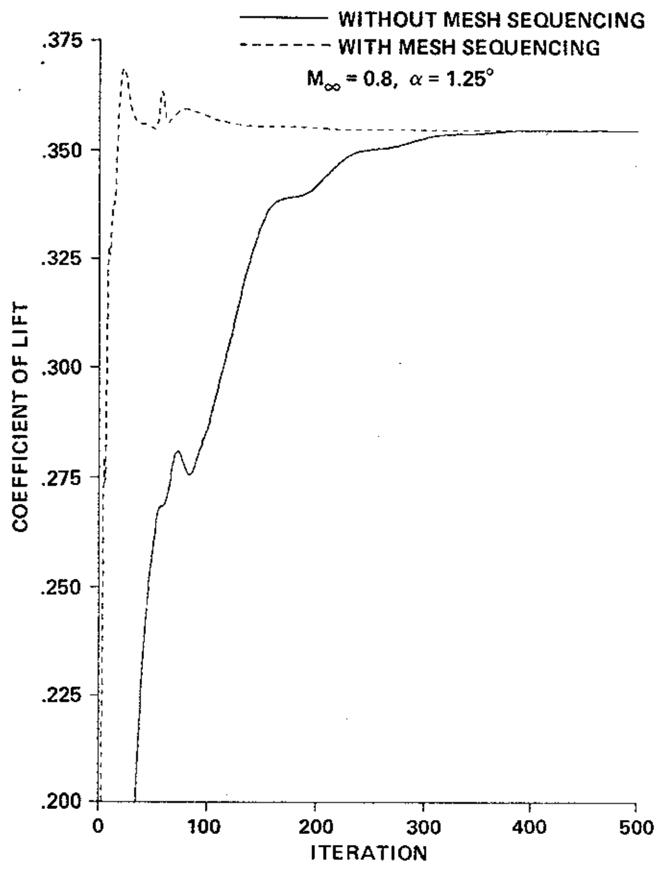


Figure 3. Improvement In Total Convergence of Lift Due to Mesh Sequencing

with T_ξ the matrix whose columns are the eigenvectors of $\widehat{A}$ and T_η the corresponding eigenvector matrix for $\widehat{B}$. They are written out in the Appendix for both two- and three-dimensions.

Replacing the Jacobians $\widehat{A}$ and $\widehat{B}$ with their eigensystem decompositions we have

$$\left[T_\xi T_\xi^{-1} + h\,\delta_\xi\left(T_\xi\,\Lambda_\xi\,T_\xi^{-1}\right)\right]\ \left[T_\eta\,T_\eta^{-1} + h\,\delta_\eta\left(T_\eta\,\Lambda_\eta\,T_\eta^{-1}\right)\right]\ \Delta Q^n = \widehat{R}^n. \tag{5.3}$$

At this point Eq. (5.2) and (5.3) are equivalent. A modified form of Eq. (5.3) can be obtained by factoring the T_ξ and T_η eigenvector matrices outside the spatial derivative terms δ_ξ and δ_η. The eigenvector matrices are functions of ξ and η and therefore this modification reduces the time accuracy to at most first-order in time. The resulting equations are

$$T_\xi\left[I + h\,\delta_\xi\,\Lambda_\xi\right]\ \widehat{N}\ \left[I + h\,\delta_\eta\,\Lambda_\eta\right]\ T_\eta^{-1}\Delta\widehat{Q}^n = \widehat{R}^n \tag{5.4}$$

where $\widehat{N} = T_\xi^{-1}T_\eta$, see the Appendix.

The explicit side of the diagonal algorithm (the steady-state finite difference equations) is exactly the same as in the original algorithm, Eq. (5.2). The modifications are restricted to the implicit side and so if the diagonal algorithm converges, the steady-state solution will be identical to one obtained with the unmodified algorithm. In fact, linear stability analysis would show that the diagonal algorithm has exactly the same unconditional stability as the original algorithm. (This is because the linear stability analysis assumes constant coefficients and diagonalizes the blocks to scalars, the diagonal algorithm then reduces to the unmodified algorithm.) The modification (pulling the eigenvector matrices outside the spatial derivatives) of the implicit operator does affect the time accuracy of the algorithm. It reduces the scheme to at most first-order in time and also gives time accurate shock calculations a nonconservative feature, i.e., errors in shock speeds and shock jumps. But, the steady-state solution is fully conservative because the steady-state equations are unmodified. Also, computational experiments have shown that the convergence and stability limits of the diagonal algorithm are essentially identical to that of the unmodified algorithm.

In two-dimensions the diagonal algorithm reduces the block tridiagonal inversion to 4 $\times$ 4 matrix multiplies and scalar tridiagonal inversions. The operation count associated with the implicit side of the full block algorithm in generalized coordinates is 410 multiplies, 356 adds, and 10 divides, a total of 776 operations, while the diagonal algorithm requires 233 multiplies, 125 adds, and 26 divides or 384 operations. Adding in the explicit side and other overhead such as I/O (input/output) and initialization, the overall savings in computational work can be as high as 40%.

In viscous calculations the diagonal algorithm can not be rigorously applied. The similarity transformations T_η and T_η^{-1} do not diagonalize the viscous Jacobian $\widehat{M}$ and therefore the η direction implicit operator of Eq. (2) cannot be reduced to diagonal form. Two approaches are used to circumvent this difficulty. One approach is to diagonalize only the ξ implicit operator and maintain the block operator in the η direction. This is not very desirable since the the reduction in computational work is lessened. The method which we actually employ in all our computations for steady viscous flow (and in convection dominated unsteady flows) is to neglect the implicit viscous term in the η operator. The implicit artificial dissipation terms provide the stabilizing factor and to date all viscous computations using this approximation have not shown any stability restriction which can be attributed to this approximation.

The algorithm in three-dimensions has the form

$$T_\xi \left[I + h\,\delta_\xi\,\Lambda_\xi\right] \,\widehat{N}\, \left[I + h\,\delta_\eta\,\Lambda_\eta\right] \widehat{P} \left[I + h\,\delta_\varsigma\,\Lambda_\varsigma\right] T_\varsigma^{-1} \Delta \widehat{Q}^n = \widehat{R}^n \tag{5.5}$$

with $\widehat{N} = T_\xi^{-1} T_\eta$ and $\widehat{P} = T_\eta^{-1} T_\varsigma$, see Appendix. Details can be found in Reference 12.

Compared to using the full block algorithm there are other significant advantages to the diagonal algorithm besides reduced computational work. One important aspect is that it simplifies coding for computer vectorization and requires less temporary storage then the block algorithm. Also, as we shall see below, the savings obtained with the diagonal form is much higher when the fourth-difference numerical dissipation is implemented implicitly.

D) Combination Smoothing

In the original algorithm given by Eq.(2) the approach of adding a constant coefficient fourth-difference explicit dissipation can produce some problems which may only be evident on refined meshes. In particular, the use of the above type of fourth-difference dissipation with a refined mesh can produce severe oscillations near shocks. In Fig. 4, a fully converged solution is shown for the NACA 0012 airfoil case at $M_\infty = 0.8$ and angle of attack, $\alpha = 1.25\,\text{deg}$. In this case the constant coefficient dissipation, Eq. (3b) was employed. A very fine grid is used in the vicinity of the shock and high frequency oscillations appear there as indicated. Varying the coefficient of artificial dissipation over a wide range did not alter the nature of this oscillation.

One can argue that the type of solution shown in Fig. 4 is quite good. It is converged, and with post processing, an excellent solution could be filtered out. In building a general purpose simulation code, however, this kind of approach is not usually practical. This is because for flows with strong shocks the oscillations can cause negative values of density with resulting numerical instability. Also,

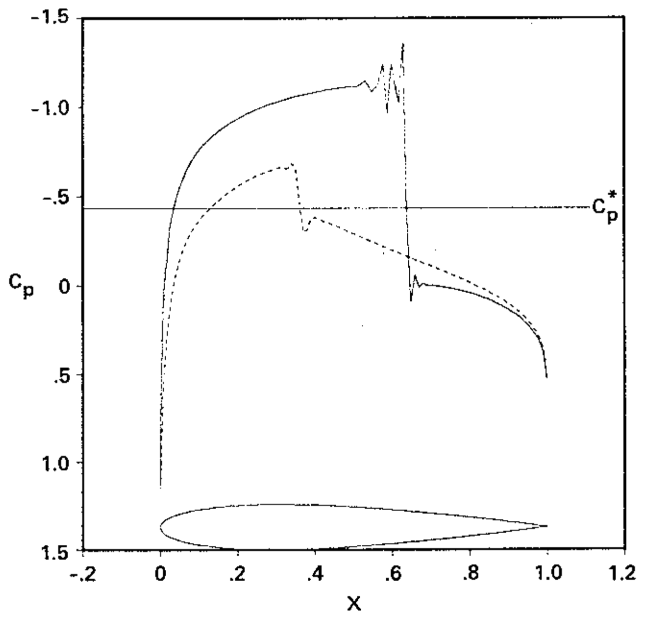


Figure 4. Coefficient of Pressure Obtained Using Fourth-difference Constant Coefficient Dissipation.

while we can sometimes converge inviscid flows with such strong oscillations, viscous flow simulations with numerically sensitive turbulence models can be unstable. Consequently, the numerical dissipation in the code has been modified to prevent such oscillations.

At shocks we generally try to locally drop the fourth-difference numerical dissipation to three point second-difference numerical dissipation. Three point second-difference numerical dissipation tend to average solution values if enough of the term can be put in because the difference equations form a diagonally dominant coefficient matrix. This can be easily seen for linearized equations because if a coefficient matrix is diagonally dominant this ensures that a solution value at a point cannot exceed the average of its neighbors. If a sufficient amount of second-difference terms can be put in, overshoots and undershoots are excluded. The problem is to determine how much of the second-difference numerical dissipation should be put in to eliminate overshoots, and, so as not to degrade overall accuracy, how to limit the application of such terms to only strong shock regions.

One way to obtain guidance as to how much numerical dissipation should be put in can be obtained by looking at upwind flux split schemes. A 3-pt second-order accurate upwind first derivative approximation can be shown (c.f. Reference 15) to be equivalent a central difference plus an added central fourth-difference numerical dissipation (much like any real matrix can be decomposed into a skew-symmetric and a symmetric matrix). This gives us a way to estimate how much numerical dissipation can be included without degrading second-order accuracy. Likewise a first-order accurate upwind scheme is equivalent to a central difference scheme plus second-difference numerical dissipation. A first-order accurate flux split scheme is essentially a monotone differencing (strong nonlinear effects can perhaps break down this statement). Insofar that the first-order upwind flux split scheme can be rewritten as a three point central difference plus second-differences, one can back out the coefficient to the dissipation that ensures monotonicity. Specifically we find

$$-\frac{1}{2\Delta\xi}(\nabla\Delta)_\xi|A|Q = -\frac{1}{2\Delta\xi}\left((|A|Q)_{j+1} - 2(|A|Q)_j + (|A|Q)_{j-1}\right)$$

should be added to $\delta_\xi E$ and so on for the other flux terms. Here $|A|$ has the usual definition of the absolute value of a matrix. Assuming A can be decomposed into its eigenvalues, $|A|$ is formed from this decomposition where the eigenvalues Λ are replaced by $|\Lambda|$. Insofar that $|A|$ is usually too complicated to form, the spectral radius of A is often used in its place, although this can overdamp some modes.

The second problem is how to isolate the second-difference numerical dissipation to just the shock. *With knowledge of the solution one can simply put such second-difference terms in over localized regions* of the flow field. But generally a more automatic approach is needed and for the most part so called product smoothers have been used (Baldwin and MacCormack[16]). That is, the coefficient of the second-difference dissipation is taken as the absolute value of a normalized second-difference of a variable such as density or pressure.

For example,

$$\Upsilon_{j,k} = \frac{|p_{j+1,k} - 2p_{j,k} + p_{j-1,k}|}{|p_{j+1,k} + 2p_{j,k} + p_{j-1,k}|}$$

where Υ is the coefficient to the second-difference dissipation at the point j, k. As a rule this coefficient will only be large near a steep gradient region, such as a shock. Work with flux limiters[17] and TVD schemes[18] is providing additional insight on how to localize the second-difference dissipation to shock regions.

The second-order product smoother can be added to the fourth-difference numerical dissipation, either throughout or in localized regions. Alternately, as the fourth-difference term itself can cause oscillations when employed across the shock, one can to switch from a fourth-difference to a second-difference, (see Jameson, et.al.[19]. A nonlinear numerical dissipation model which works very well in general cases has the form

$$\nabla_\xi\left(\sigma_{j+1,k}J^{-1}_{j+1,k} + \sigma_{j,k}J^{-1}_{j,k}\right)\left(\epsilon^{(2)}_{j,k}\Delta_\xi Q_{j,k} - \epsilon^{(4)}_{j,k}\Delta_\xi\nabla_\xi\Delta_\xi Q_{j,k}\right) \tag{7}$$

with

$$\begin{aligned}\epsilon^{(2)}_{j,k} &= \kappa_2\Delta t\, f(\Upsilon_{j+1,k}, \Upsilon_{j,k}, \Upsilon_{j-1,k})\\ \epsilon^{(4)}_{j,k} &= \max(0, \kappa_4\Delta t - \epsilon^{(2)}_{j,k})\end{aligned}$$

where typical values of the constants are $\kappa_2 = 1/4$ and $\kappa_4 = 1/100$ and $f(\Upsilon)$ is some smooth function over the domain of interest. Similar terms are used in the η and ς directions. The coefficient $\sigma_{j,k}$ is a spectral radius scaling and in two-dimensions is defined as

$$\sigma_{j,k} = |U| + a\sqrt{\xi_x^2 + \xi_y^2} + |V| + a\sqrt{\eta_x^2 + \eta_y^2}$$

which is a sum of the spectral radii of $\widehat{A}$ and $\widehat{B}$.

The first part of Eq. (7) is a second-difference dissipation with an extra pressure gradient coefficient to increase its value near shocks. The second part is a fourth-difference where the logic to compute $\epsilon^{(4)}_{j,k}$ switches it off when the second-difference nonlinear coefficient is larger then the constant fourth-difference coefficient. This occurs right near a shock. This model is the current artificial dissipation model used in our implicit scheme.

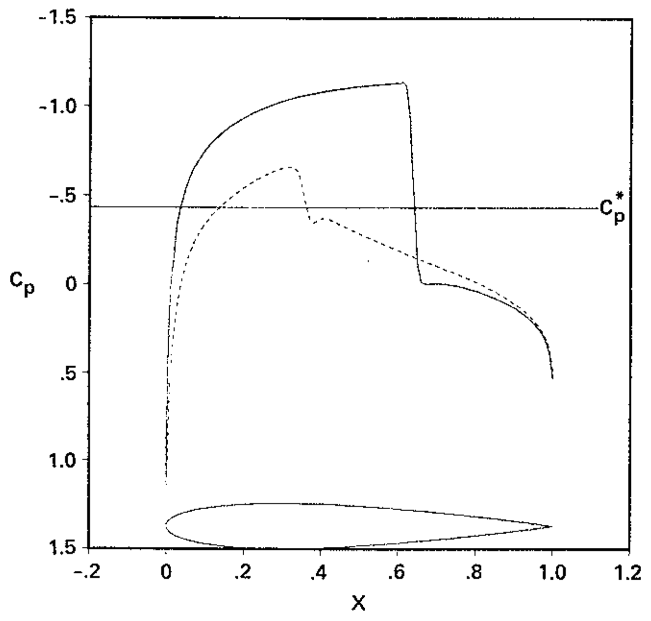


Figure 5. Coefficient of Pressure Obtained Using Nonlinear Dissipation.

The explicit dissipation model, Eq. (7), is added to the right hand side of the algorithm in place of Eq. (3a). At this point the original second-difference implicit constant coefficient dissipation is retained to stabilize the explicit terms. In Fig. 5 we show solutions for the flow problem of Fig. 4, using this explicit nonlinear artificial dissipation. The oscillations at the shock are eliminated and a sharp shock is obtained. The results shown are fully converged to machine zero. The chosen values of the coefficients have, at least to date, been static and are not changed from case to case.

E) Implicit Smoothing

If the numerical algorithm given by Eq.(2) were unfactored, fully implicit in boundary condition and numerical dissipation treatment, then for $\Delta t \to \infty$ it would essentially represent a Newton iterative method and very few iterations would be needed to achieve a steady state. Of course, leaving the left-hand-side operator of Eq.(2) unfactored would necessitate a very costly matrix solution process with very high storage requirements and so approximate factorization is used. As a rule though, the more implicit and unfactored the algorithm, the better the iterative convergence rate will be.

Ideally the fourth-difference numerical dissipation operators should be treated implicitly. However, including this term implicitly increases the band structure of each implicit operator to pentadiagonal and, although the outer bands are diagonal, their elimination significantly increases the inversion work and storage requirements. If the artificial dissipation terms are put in only explicitly ($\epsilon_i = 0$), then stability is maintained only for small Δt. It is for this reason that the numerical dissipation terms D_i are included implicitly. Because only second-difference terms are treated implicitly, the matrix band structure is not increased, but this is not a correct implicit treatment of the artificial dissipation, and to maintain unconditional stability, ϵ_i must be greater than $2\epsilon_e$. Generally for nonlinear problems $\epsilon_i = 2.5\epsilon_e$. While choice of ϵ_i maintains stability, analysis of a model problem[15] shows that the

ERRATA for AIAA Paper 85-0360

Recent Improvements in Efficiency, Accuracy, and Convergence
for Implicit Approximate Factorization Algorithms

Thomas H. Pulliam and Joseph L. Steger

page 6, Figure 2 : Lables VARIABLE Δt and CONSTANT Δt are incorrect and should read

VARIABLE Δt	———
CONSTANT Δt	- - - - - - -

page 17, Figure 8 : Angle of attack for $M_\infty = 0.63$ should be $\alpha = 2.0°$

page 24, Table 3: Entry for C_D in the second to last row should be 0.0089 convergence rate is adversely effected compared to a correct fully implicit treatment of the numerical dissipation.

Although a fully implicit treatment of the fourth-difference numerical dissipation would significantly increase the computational work and storage requirements for inversion of the full block algorithm, the cost of pentadiagonal inversion is greatly reduced if the diagonal form of the algorithm is used. This is because only scalar pentadiagonals are needed for the diagonal form. In fact, the computational cost of solving pentadiagonals in the diagonal algorithm is even less than the original block tridiagonal system. The net effect is enhanced convergence and stability with a minimal increase in cost.

Treating the fourth-difference numerical dissipation operators implicitly in the diagonal algorithm gives us the altered form

$$T_\xi \left[I + h\delta_\xi \Lambda_\xi - hD_i|_\xi\right] \widehat{N} \left[I + h\delta_\eta \Lambda_\eta - hD_i|_\eta\right] \widehat{P} \left[I + h\delta_\zeta \Lambda_\zeta - hD_i|_\zeta\right] T_\zeta^{-1} \Delta \widehat{Q}^n = \widehat{R}^n \tag{8}$$

where the D_i are redefined as

$$D_i|_\xi = \nabla_\xi \left(\sigma_{j+1,k} J_{j+1,k}^{-1} + \sigma_{j,k} J_{j,k}^{-1}\right) \left(\epsilon_{j,k}^{(2)} \Delta_\xi - \epsilon_{j,k}^{(4)} \Delta_\xi \nabla_\xi \Delta_\xi\right) J$$

Terms for $D_i|_\eta$ and $D_i|_\zeta$ have a similar form.

The effectiveness of treating the fourth-difference terms implicitly in the NACA 0012 test case is indicated by Fig. 6. Shown here are results using the explicit nonlinear dissipation, Eq. (7), with implicit second-difference constant coefficient dissipation and with the fully implicit treatment, Eq.(8). Full implicit treatment of the numerical dissipation improves convergence and also improves the robustness of the code. In general, the practical stability limits of the codes are increased with implicit treatment of the dissipations.

IV) BOUNDARY CONDITIONS

Boundary conditions uniquely define the solution. Given the the same geometry and conditions at boundaries a wide variety of algorithms should all give a consistent set of results. This usually depends on the manner in which boundary conditions are implemented. This implementation takes two important forms: the physical conditions (for instance fixed total pressure at an upstream boundary or tangency at an inviscid surface) and the numerical boundary conditions (including the extra conditions which may arise from numerics such as happens with central differencing and also the numerics involved in applying the physical conditions). It is important that the boundary conditions used in a computation be well defined and reported on in detail otherwise cross comparisons among different computations are difficult to access.

A particular set of boundary conditions employed in airfoil computations are described below in detail. The geometry is mapped onto the computational rectangle such that all the boundary surfaces are edges of the rectangle, see Fig. 7 . This application is for a "C" mesh topology. In "O" mesh topologies the wake cut boundary is periodic and can be handled as such where periodic solvers are used in the implicit inversions.

A. Body Surfaces

At a rigid body surface, tangency must be satisfied for inviscid flow and the no slip condition for viscous flow. In two-dimensions body surfaces are usually mapped to η = constant coordinates. The normal component of velocity in terms of the curvilinear metrics is given by

$$V_n = \frac{\eta_x u + \eta_y v}{\sqrt{(\eta_x^2 + \eta_y^2)}} \tag{9.1}$$

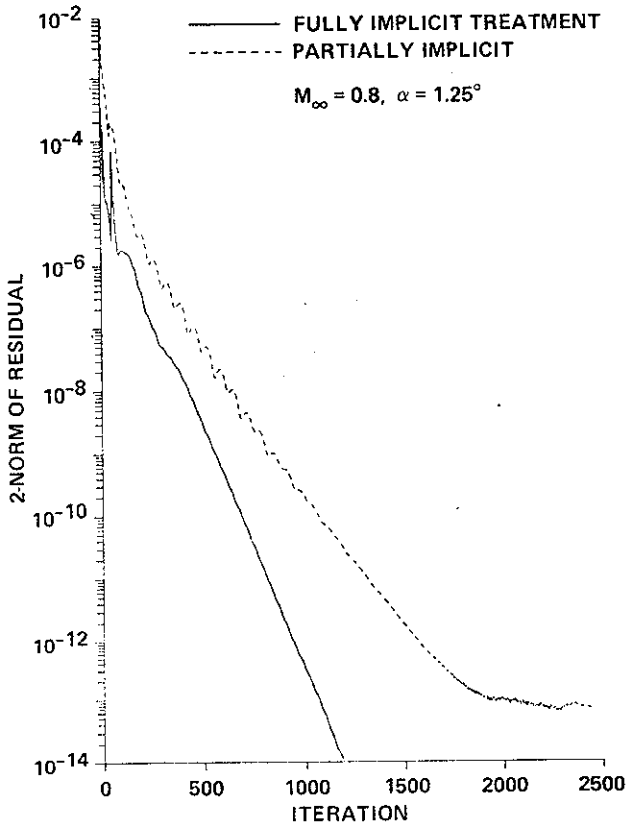


Figure 6. Improvement in Convergence Due to Implicit Treatment of Nonlinear Dissipation Terms.

and the tangential component is

$$V_t = \frac{\eta_y u - \eta_x v}{\sqrt{(\eta_x^2 + \eta_y^2)}} \tag{9.2}$$

Therefore, tangency is satisfied by $V_n = 0$ (no flow through the body). The tangential velocity V_t is obtained at the body surface through linear extrapolation for inviscid cases and is set to zero for viscous cases. The Cartesian velocities are then formed from the inverse relation between them and Eq. (9) where

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1.}{\sqrt{(\eta_x^2 + \eta_y^2)}} \begin{bmatrix} \eta_y & -\eta_x \\ \eta_x & \eta_y \end{bmatrix} \begin{pmatrix} V_t \\ V_n \end{pmatrix}. \tag{10}$$

The extrapolation of V_t produces less error if the mesh lines are clustered to the body surface. The velocities of Eqs. (9.1), and (9.2) are scaled such that the metric variations are removed which decreases the errors in the extrapolations especially for nonorthogonal meshes.

The pressure on the body surface is obtained from the normal momentum equation

$$\begin{aligned} \rho[\partial_\tau \eta_t + u\partial_\tau \eta_x + v\partial_\tau \eta_y] - \rho U(\eta_x u_\xi + \eta_y v_\xi) = \\ (\eta_x \xi_x + \xi_y \eta_y) p_\xi + ({\eta_x}^2 + {\eta_y}^2) p_\eta = p_n \sqrt{{\eta_x}^2 + {\eta_y}^2} \end{aligned} \tag{11}$$

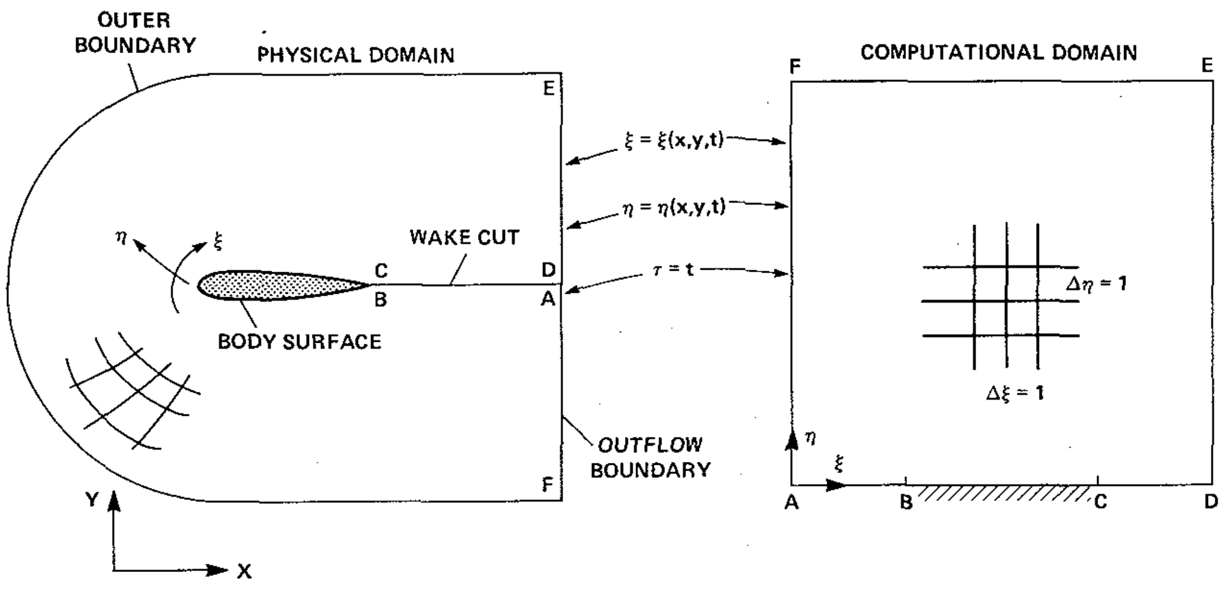


Figure 7. Topological Mapping of an Airfoil onto a "C" Mesh.

where n is the local normal to the body surface. Equation (11) is solved at the surface using central second-order accurate differences in ξ and one-sided first- or second-order accurate differences in η. For steady uniform incoming flow free-stream stagnation enthalpy H_0 is held constant along the body in inviscid flow. Using the equation for enthalpy $H_0 = (e+p)/\rho$ and the computed velocities and pressure, a value of density is obtained at the body. Adiabatic or constant temperature walls are used for viscous and unsteady flows to obtain density at the surface. In either case, total energy e is decoded from the equation of state.

B. Free Surfaces

Stretched grids are usually used to place far field boundaries far away from body surfaces. When bow shocks and attached shocks are generated at a body surface care is taken to ensure that the shocks are sufficiently weak when they reach far field boundaries so that they are not reflected or at least they reflect outside the flow domain. A nonreflective characteristic like boundary procedure is used at far field boundaries.

For subsonic free stream locally one-dimensional Riemann invariants are used at the outer far field boundaries. The locally one-dimensional Riemann invariants are given in terms of the normal velocity component as

$$R_1 = V_n - 2a/(\gamma - 1) \quad \text{and} \quad R_2 = V_n + 2a/(\gamma - 1) \tag{12}$$

The Riemann invariants R_1, R_2 are associated with the two characteristic velocities (locally one-dimensional) $\lambda_1 = V_n - a$ and $\lambda_2 = V_n + a$ respectively. Two other equations are needed so that four unknowns (the four flow variables) can be calculated. We choose V_t and $S = \ln(p/\rho^\gamma)$ where S is entropy. At the far field boundaries shown in Fig. 7, the normal n is directed away from the boundary. For subsonic inflow $V_n < 0$ and the characteristic velocity $\lambda_1 < 0$, therefore the characteristic variable R_1 can be specified along with two other conditions. The Riemann invariant R_1, V_t and S are all set to free stream values. The other characteristic velocity $\lambda_2 > 0$ and R_2 is extrapolated from the interior flow variables. On subsonic outflow $V_n > 0$ and $\lambda_2 > 0$ while $\lambda_1 < 0$ so only R_1 is fixed to free stream and R_2, V_t and $\ln(S)$ are extrapolated. Once these four variables are available at the boundary the four flow variables Q can be obtained. For supersonic inflow boundaries all flow variables are specified and for supersonic outflow all variables are extrapolated.

Along singularities or cuts in the geometry (such as the wake cut in a "C" mesh), averaging is used to provide continuous flow variables. As mentioned above periodic conditions are used for "O" meshes.

C. Far Field Circulation Correction

For lifting airfoils in subsonic free stream, circulation at the far field boundary is accounted for to first-order (following Salas, et. al.[20]) by imposing a compressible potential vortex solution which is added as a perturbation to the free stream quantities ($u_\infty = M_\infty \cos(\alpha)$ and $v_\infty = M_\infty \sin(\alpha)$). The perturbed far field boundary velocities are defined as

$$u_f = u_\infty + \frac{\beta \Gamma \sin(\theta)}{2\pi r \left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)} \tag{13.1}$$

and

$$v_f = v_\infty - \frac{\beta \Gamma \cos(\theta)}{2\pi r \left(1 - M_\infty^2 \sin^2(\theta - \alpha)\right)} \tag{13.2}$$

where the circulation $\Gamma = \frac{1}{2} M_\infty l C_l$, l is the chord length, C_l the coefficient of lift at the surface, M_∞ the free stream Mach number, α the angle of attack, $\beta = \sqrt{1 - M_\infty^2}$ and r, θ are polar coordinates to the point of application on the outer boundary relative to an origin at the quarter chord point on the airfoil center line. A corrected speed of sound is also used which enforces constant free stream enthalpy at the boundary where

$$a_f^2 = (\gamma - 1)\left(H_\infty - \frac{1}{2}(u_f^2 + v_f^2)\right) \tag{13.3}$$

Equations (13) are used instead of free stream values in defining the fixed quantities for the far field characteristic boundary conditions to be consistent with the surface lift.

Figure 8 shows the coefficient of lift C_l plotted against the inverse of the distance to the outer boundary for a NACA 0012 airfoil at the transonic condition $M_\infty = 0.8$, $\alpha = 1.25\,\text{deg}$ and at subcritical conditions $M_\infty = 0.63$, $\alpha = 2.0\,\text{deg}$. In these cases the outer boundary varies for 4.5 chords to 96 chords where outer mesh rings were eliminated from the 96 chord grid to produce the cut down meshes. This insures that the grid spacing between the body and outer boundary is identical for all the cases. Without the far field vortex correction the lift of the subcritical case can vary by as much as 12 % as seen in Fig. 8. With the far field vortex logic the subcritical case now has virtually no variation with outer boundary distance. For the transonic case we see roughly a 1 - 2 % change which is quite good considering the strength of the shocks. The typical distance chosen for most cases presented here is 25 chords.

The vortex correction logic can be modified to produce boundary conditions which allow one to compute the angle of attack for a given lift. This is done by fixing the circulation Γ in Eq. (13) at its value for the given lift. An iterative procedure is used where the lift computed at the surface is compared to the desired lift and then the initial angle of attack is modified by the formula

$$\Delta\alpha = -\beta_\alpha \left(C_l(input) - C_l(calculated)\right)$$

with β_α a relaxation parameter on the order of 2 . Computations in which a specified lift resulted in an angle of attack were compared with fixed α solutions at the same Mach number and showed excellent agreement. This procedure has been verified in numerious numerical examples.

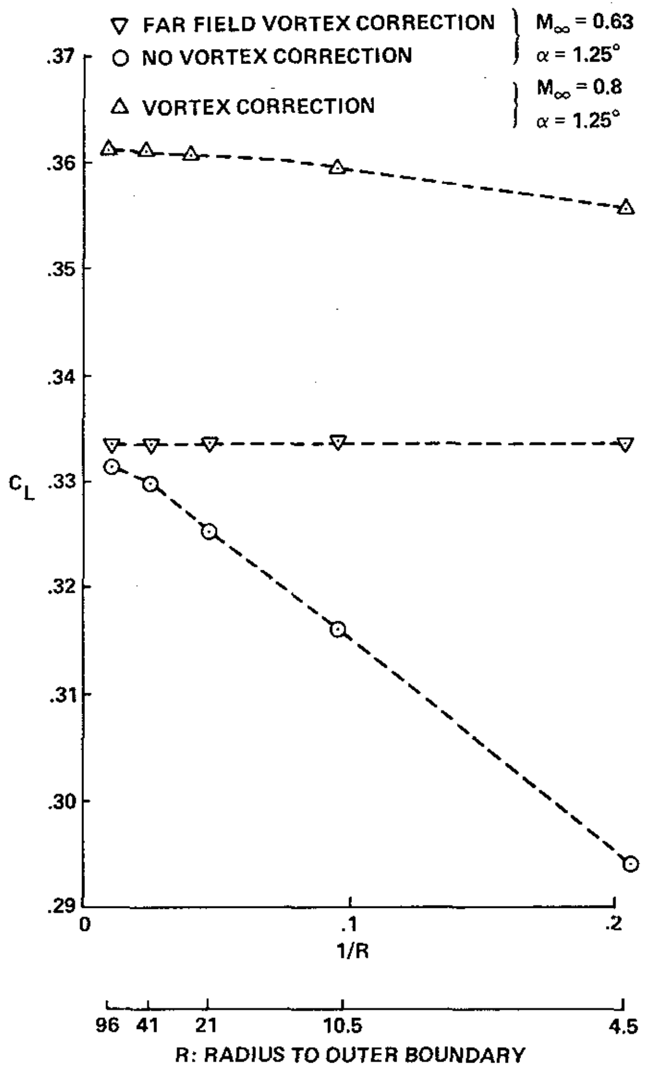


Figure 8. Effect on Lift of Varying Outer Boundary Distances With and Without Vortex Correction.

V) COMPARISONS

The ARC codes and their derivatives have been applied to steady and unsteady airfoils, cascades, projectiles, inlets, base flow, blunt body problems, and various three-dimensional simulations about wings and bodies. To demonstrate the increased efficiency of the new algorithmic changes we will only look at two classes of flow ; steady transonic inviscid and viscous flow about a single element airfoil and steady flow about a 3-D ellipsoid at angle of attack.

A) AIRFOIL

The grid used for the NACA 0012 test case is an "O" mesh topology with 192 points on the airfoil surface (running from the lower trailing edge around the nose to the upper trailing edge) and 33 point in the normal direction. The grid is clustered at the leading and trailing edges, near the expected shock locations on the upper and lower surfaces, and in the normal direction as shown in Fig. 9.

Results obtained from ARC2D are shown in Fig. 10 in the form of coefficient of pressure, Mach contours, and pressure contours. These are identical to the ones shown in Fig. 1, but are repeated here to scale for comparison with results from Jameson's multigrid Euler code FLO52R[19] as shown in Fig. 11. This is an Euler code which uses a multistage Runge-Kutta like algorithm with a multigrid scheme to accelerate convergence. The code employs enthalpy damping, residual averaging and the nonlinear artificial dissipation model discussed previously. The two codes were run on the same machine, the CRAY XMP at NASA Ames, on the same meshes and at the same flow conditions. The comparison between the two codes is quite good, despite the differences in the spatial discretion method.

A number of convergence criteria have been chosen to assess the efficiency and convergence rates of the codes. Computer times are chosen as the measure of relative speed. Since the two codes are run on the same machines and with the same meshes this is an adequate measure. Other measures such as operation count, work or iteration count are usually programming dependent or susceptible to misinterpretation. The convergence criteria used here are:

1. Coefficient of lift (C_L) to 1% of converged value.
2. Coefficient of lift (C_L) to 1/2% of converged value.
3. Coefficient of lift (C_L) to 5 decimal places.
4. Number of supersonic points to converged value.
5. Residual to machine zero. (10^{-13} on the Cray XMP.)

The residual is the l_2 norm of the explicit or right hand side of Eq.(2). We use just the component from the continuity equation, the other components behave similarly. For the above case on the 192 by 33 mesh the computer times (in CPU seconds) for the convergence criteria are given in Table 1.

Convergence Comparison (seconds)		
Criteria	ARC2D	FLO52R
1% of C_L	6	8
1/2% of C_L	17	10.5
C_L to 5 places	57	31
No. S.S. pts	36	17
Machine zero	120	97

Table 1. Convergence Data for 192 by 33 grid.

To investigate the effect of mesh refinement a grid of 248 by 49 points is employed as the second study. The mesh is refined more at the nose, tail and near the shocks as shown in Fig. 12.

Computational results for ARC2D and FLO52R are shown in Figs. 13 and 14. In this case the shocks are sharper. The computed convergence data for this case is contained in Table 2. In Figure 15, we show convergence history versus iteration for the two ARC2D results.

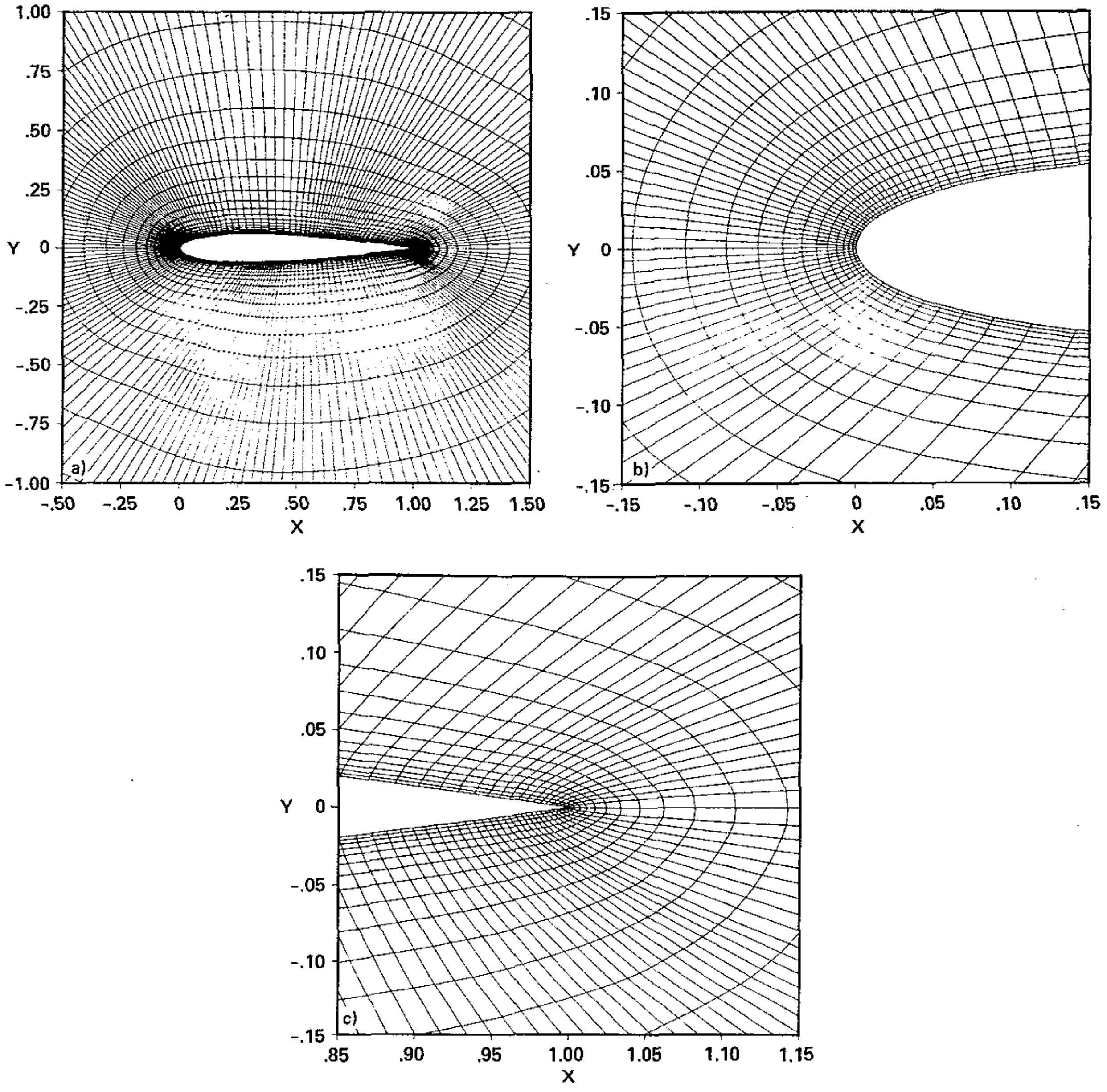


Figure 9. NACA0012 Grid Using 192 by 33 Grid Points.

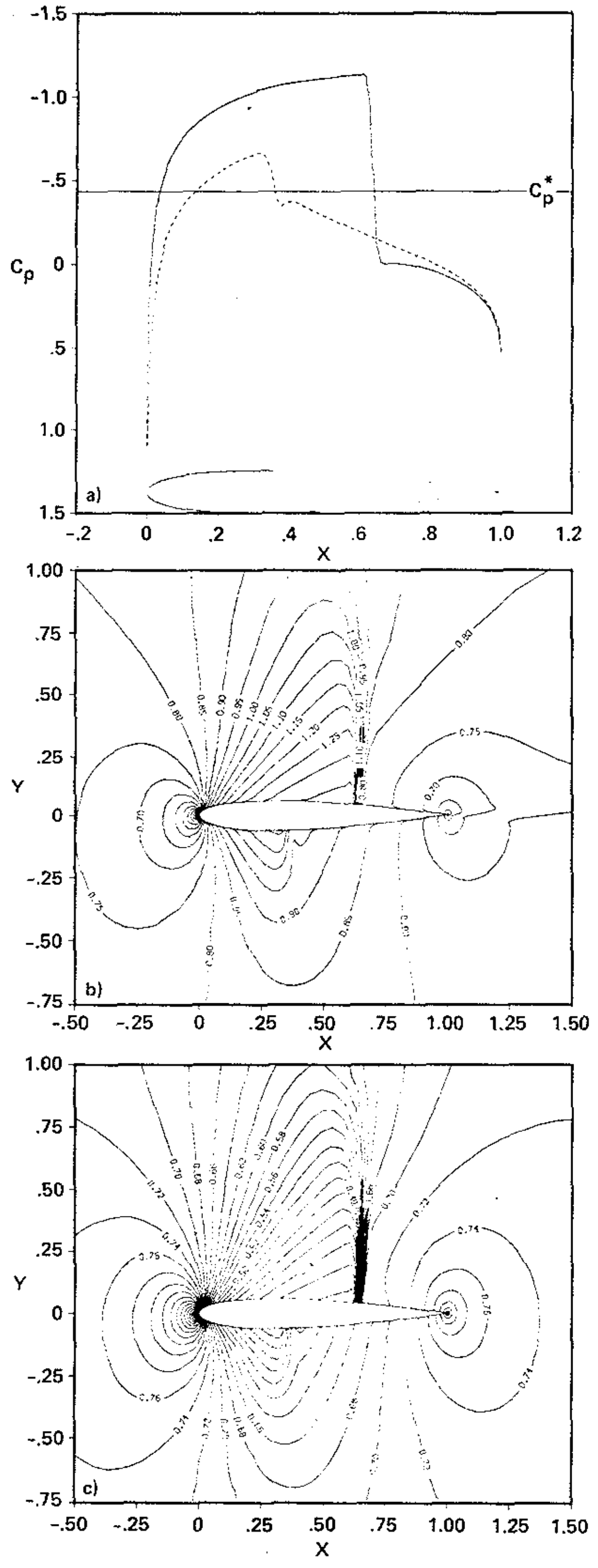


Figure 10. ARC2D Results.

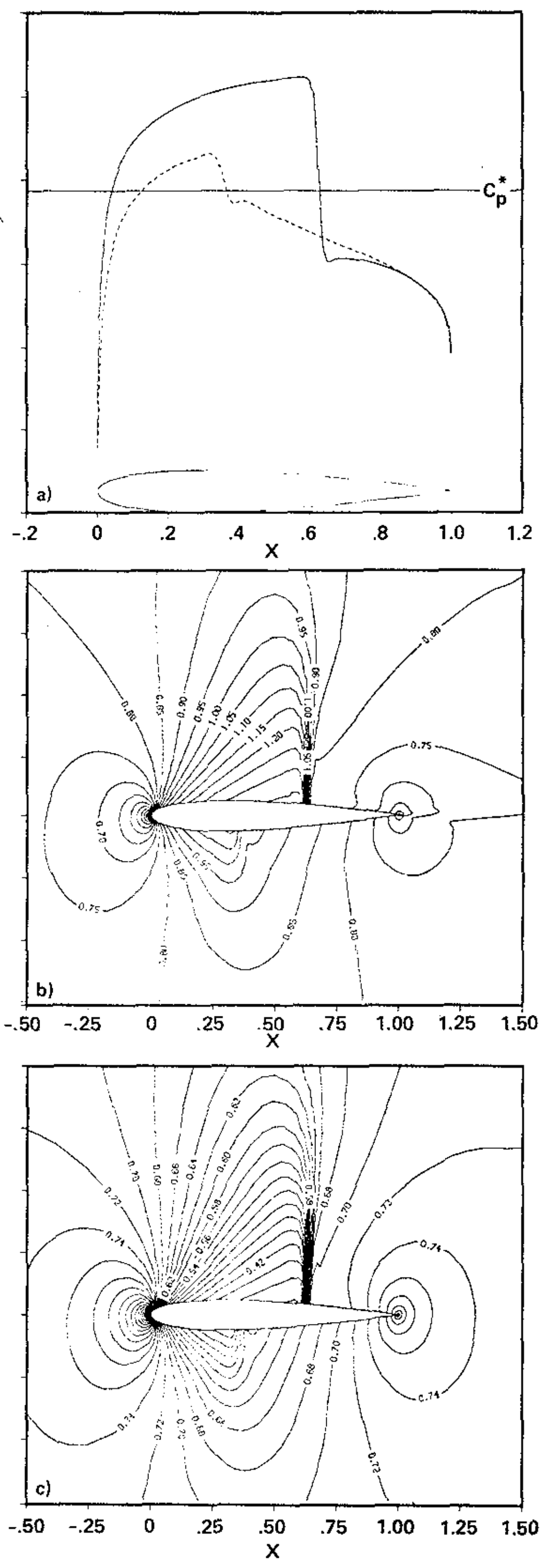
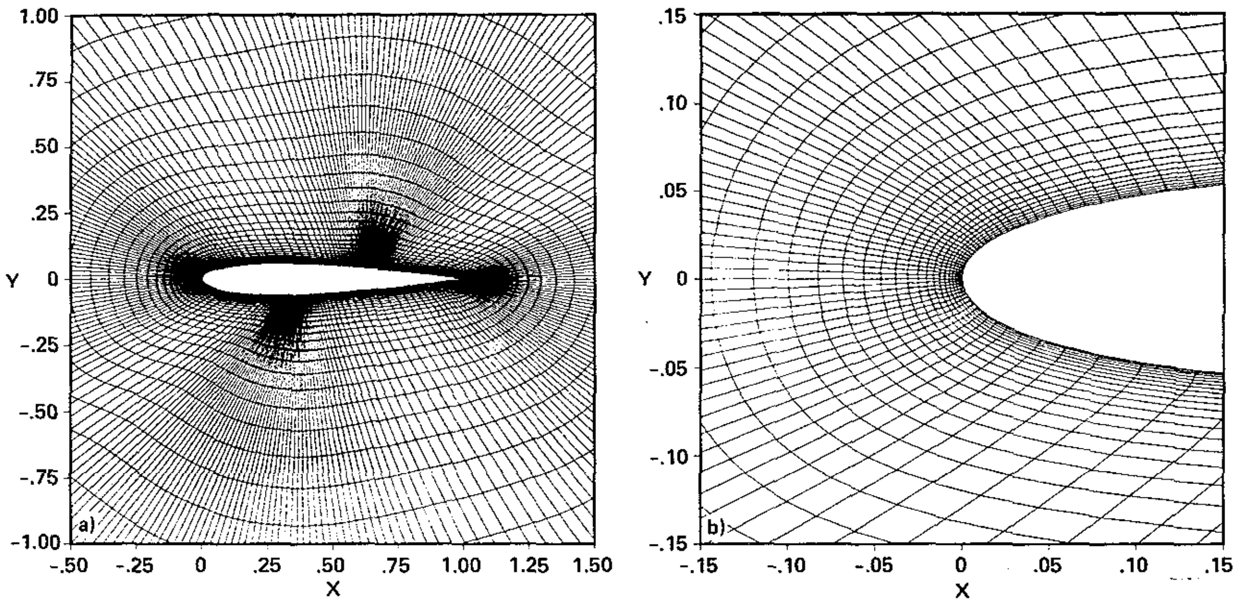


Figure 11. FLO52R Results.



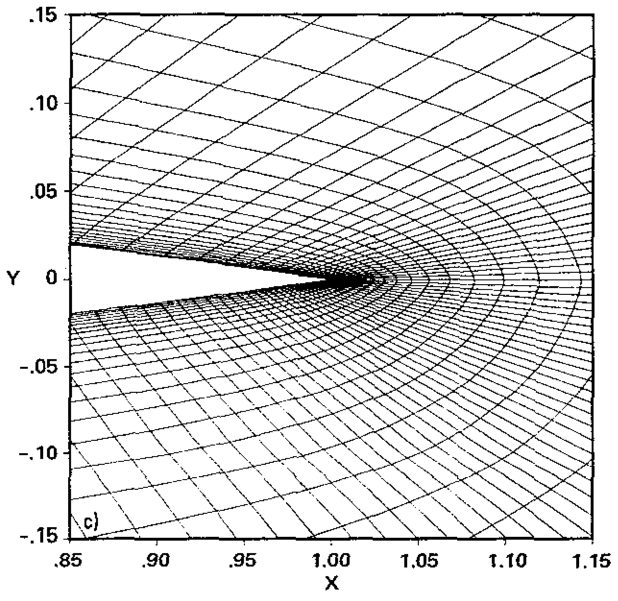


Figure 12. NACA0012 Mesh, 248 by 49.

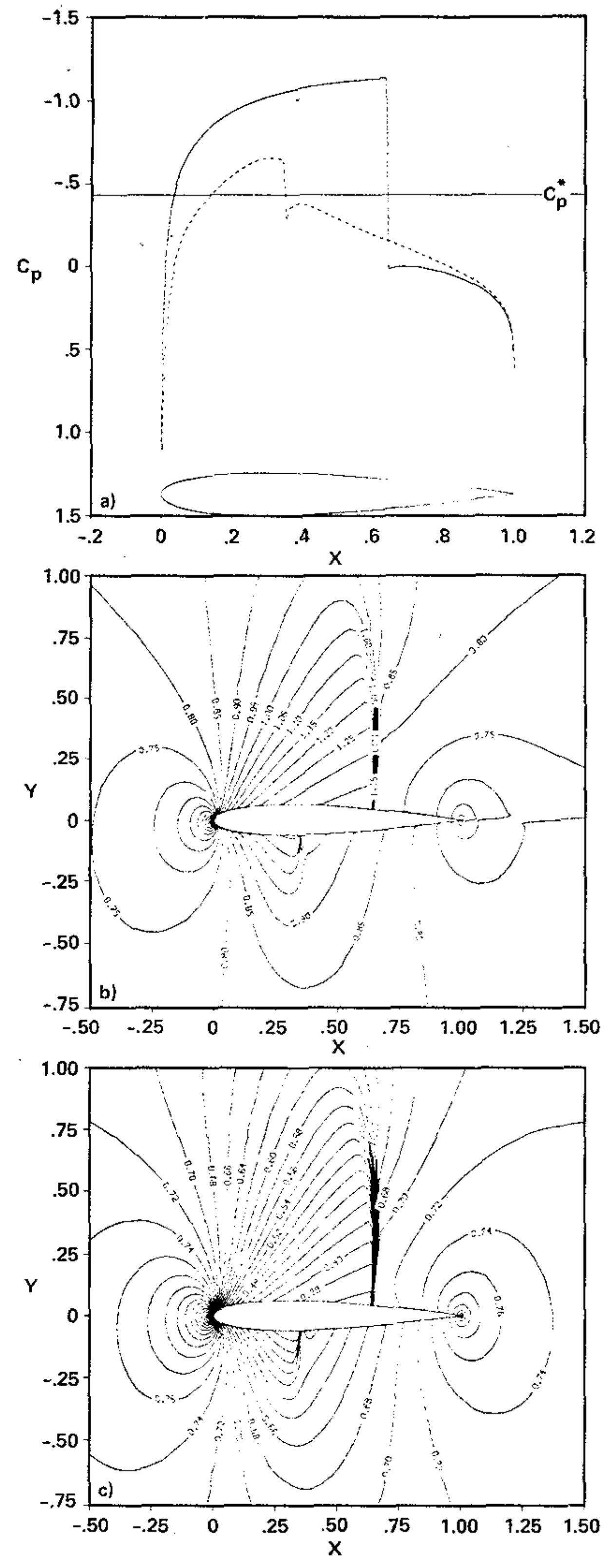


Figure 13. ARC2D Fine Grid Results

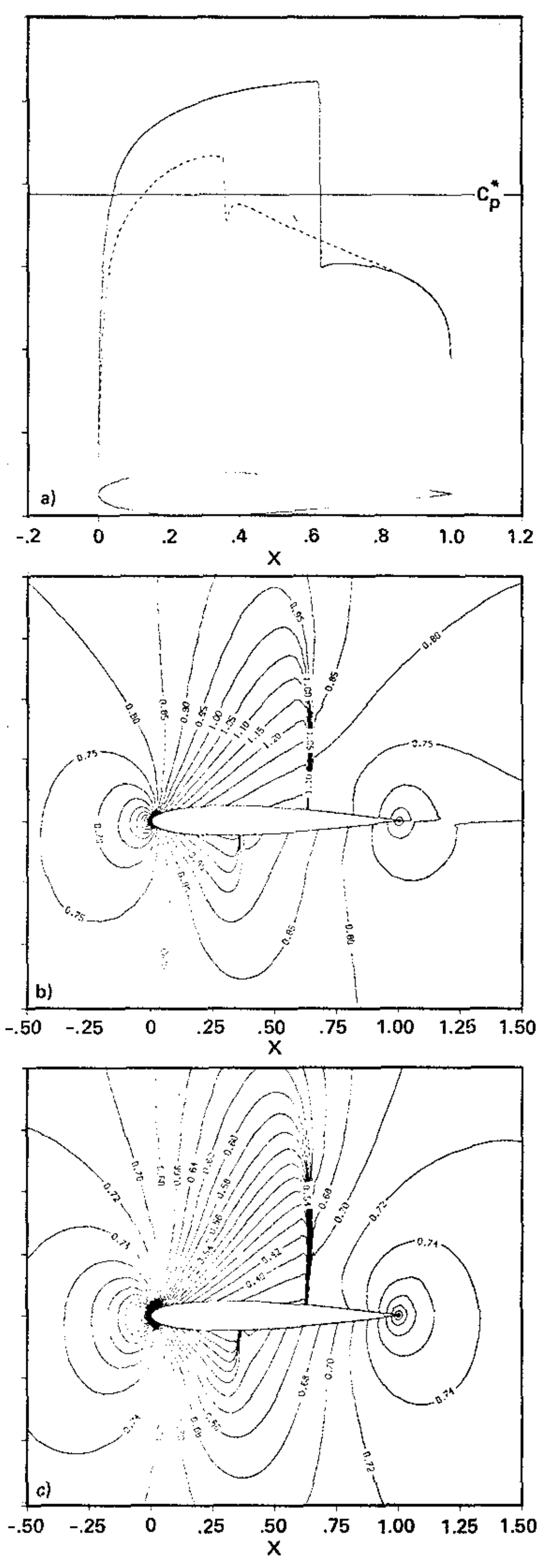


Figure 14. FLO52R Fine Grid Results

Convergence Comparison (seconds)		
Criteria	ARC2D	FLO52R
1% of C_L	38	23
1/2% of C_L	52.5	25.5
C_L to 5 places	174	168.5
No. S.S. pts	118	160
Machine zero	376	800+

Table 2. Convergence Data for 248 by 49 grid.

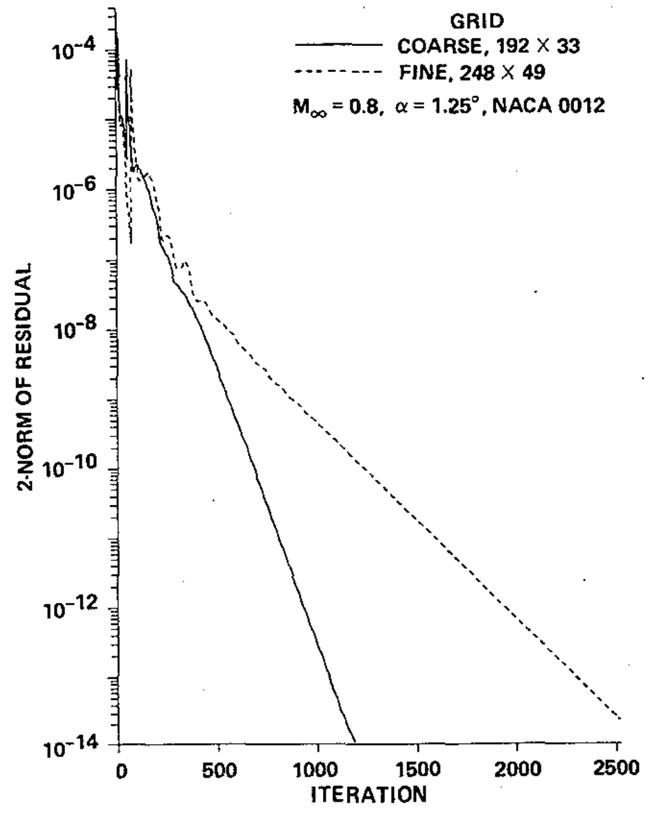


Figure 15. Convergence History verses Iteration for ARC2D Results.

The algorithm was applied to viscous flow as well. Two example cases are presented below. The cases are taken from the suggested problems of the 1980-81 AFOSR-HTTM-Stanford Conference on Complex Turbulent Flows[21] an RAE2822 airfoil at $M_\infty = 0.676$, $\alpha = 1.93\,\text{deg}$, $Re = 5.7 \times 10^6$ and $M_\infty = 0.73$, $\alpha = 2.79$ and $Re = 6.5 \times 10^6$.

Results obtained from ARC2D for the first case are shown in Fig. 16. The grid used is a 248 by 51 point "O" mesh. An algebraic turbulence model (Baldwin and Lomax[22]) was used and transition was fixed at 11% chord. Experimental data due to Cook et. al.[23] is used for comparison. We see a good comparison with experiment for pressure coefficient and some boundary layer quantities.

The computed lift, drag and moment are compared with other computations and the experiment in Table 3. Due to the uncertainty of the angle of attack correction all computors matched lift. We show here our computation for both the experimentally corrected angle of attack and the values when lift is matched using the procedure outline above. Also shown are results from computors at the Stanford Conference and some recent results of Mehta[24]. The overall comparison with experiment and other computations is quite good.

Loads – RAE2822 Airfoil – $M_\infty = 0.676, Re = 5.7 \times 10^6$						
	α	C_L	C_{DP}	C_{Df}	C_D	C_M
Experiment	2.40	0.566			0.0085	-0.082
Corrected Exp.	1.93	0.566			0.0085	-0.082
Mehta (1983)	1.80	0.566	0.0033	0.0061	0.0094	-0.087
Melnik (1981)	1.84	0.566	0.0027	0.0060	0.0087	-0.082
Le Balleur (1981)	1.93	0.566	0.0036	0.0056	0.0092	-0.080
Present	1.93	0.576	0.0034	0.0055	0.089	-0.081
Present Cor. α	1.87	0.566	0.0034	0.0055	0.0089	-0.081

Table 3. Forces for RAE2822 Viscous Calculation.

Results obtained for the second case are shown in Fig. 17. The grid used is a 248 by 51 point "O" mesh. The turbulence model was used and transition was fixed at 3% chord. We again see a good comparison with experiment for pressure coefficient and boundary layer quantities.

The computed lift, drag and moment are compared with other computations and the experiment in Table 4. Results from computors at the Stanford Conference[21] and results of Mehta[24] are shown. Again lift is matched using the procedure presented in Section IV-C. The overall comparison with experiment and other computations is again quite good. The shock location on the upper surface compares well. In the present computations a small region of separated flow occurs at the base of the shock and near the trailing edge on the upper surface. Residual convergence history versus iteration for these cases are shown in Fig. 18. Table 5 shows the computed convergence criteria data.

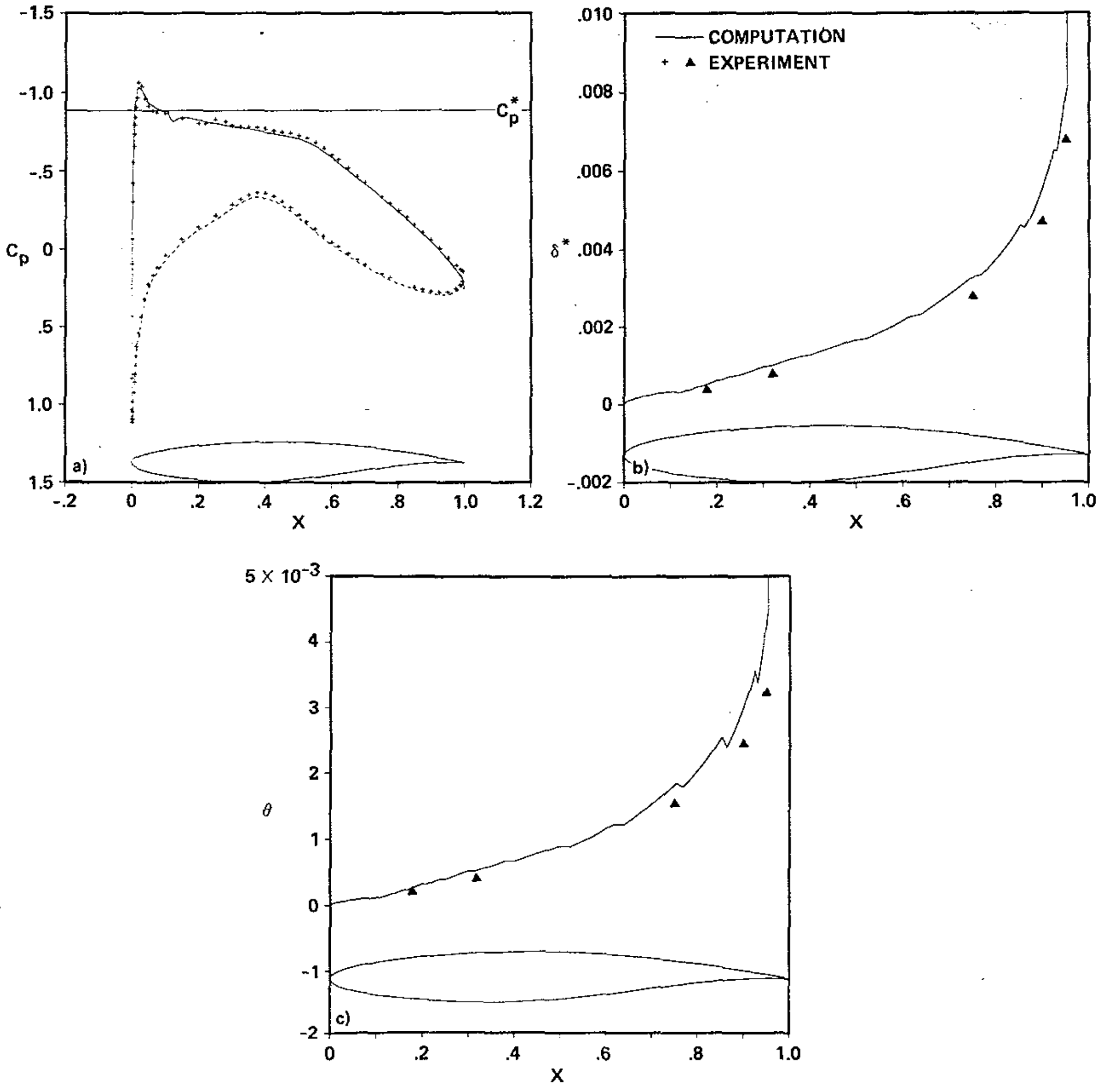


Figure 16. Viscous Results for RAE2822 Airfoil at $Re = 5.7 \times 10^6$, $M_\infty = 0.676$, $\alpha = 1.93\,\text{deg}$.

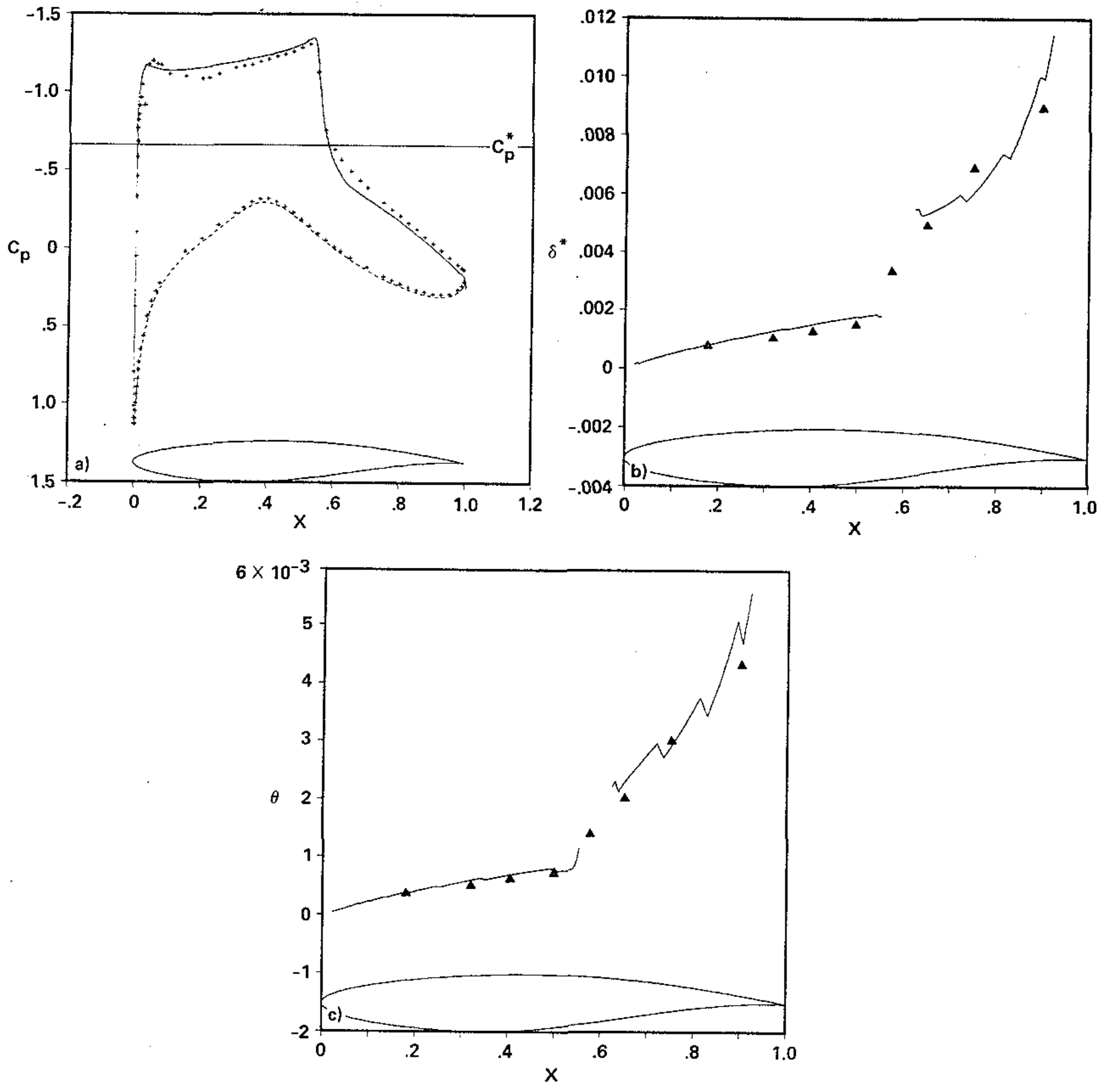


Figure 17. Viscous Results for RAE2822 Airfoil at $Re = 6.5 \times 10^6$, $M_\infty = 0.73$, $\alpha = 2.79$ deg.

Loads - RAE2822 Airfoil - $M_\infty = 0.73, Re = 6.5 \times 10^6$						
	α	C_L	C_{DP}	C_{Df}	C_D	C_M
Experiment	3.19	0.803			0.0168	-0.099
Corrected Exp.	2.79	0.803			0.0168	-0.099
Mehta (1983)	2.79	0.793	0.0118	0.0059	0.0177	-0.094
Melnik (1981)	2.54	0.803	0.0100	0.0057	0.0157	-0.094
Le Balleur (1981)	2.79	0.787	0.0111	0.0055	0.0166	-0.086
Present	2.79	0.824	0.0128	0.0050	0.0178	-0.092
Present Cor. α	2.67	0.803	0.0113	0.0051	0.0164	-0.092

Table 4. Forces for RAE2822 Viscous Calculation.

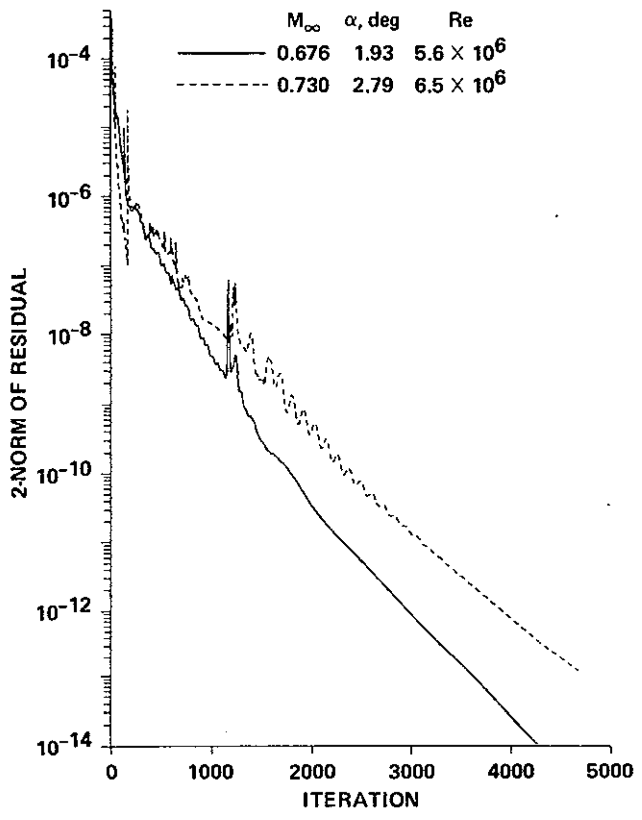


Figure 18. Convergence History for RAE2822 Viscous Cases.

Convergence Comparison (seconds)		
.Criteria	$M_\infty = 0.676$	$M_\infty = 0.73$
1% of C_L	212	191
1/2% of C_L	264	295
C_L to 5 places	712	738
No. S.S. pts	608	513
Machine zero	1147	1203

Table 5. Convergence Data RAE2822 Viscous Cases.

B) ELLIPSOID

Application of a number of the algorithmic changes into the three-dimensional code (ARC3D) has greatly improved its accuracy, efficiency, convergence and robustness. The code is highly vectorized, uses the diagonal algorithm, the nonlinear explicit and implicit artificial dissipation and the spatially varying time step. In the original code developed in 1978[2] the convergence rate (reduction in residual per iteration) for typical cases was on the order of 0.996, for the current algorithm it has been reduced to approximately 0.986. The computation time for 27000 grid points has been reduced from about 400 minutes for a converged case on the CDC7600 to about 5 minutes on a CRAY-XMP. These numbers can be improved upon since not all of the algorithmic and coding modifications have been implemented.

An ellipsoid at angle of attack has been selected as a three-dimensional test case. Figures 19 to 25 illustrate solutions obtained for a 6-1 three-dimensional ellipsoid at $M_\infty = 0.74$ and an angle of attack $\alpha = 25$ degrees. The Reynolds number is 44×10^6 and the flow is turbulent. A bilateral symmetry condition is used to reduce the solution domain. The grid used employs 40 points along the surface, in the streamwise direction, 21 points circumferentially from windward to leeward side, and 30 points in the radial direction. The surface definition is shown in Fig. 19 and Fig. 20 shows some of the grid definition at the symmetry plane and surface. Various views showing three-dimensional particle paths (passive particles traced out in a Lagrangian manner using the velocity field) indicate massive separation, as seen in Figs. 21,22, and 23. Simulated oil flow patterns shown in Figs. 24 and 25 indicates surface separation lines. These results have been obtained on a coarse grid and should be considered preliminary. A more detailed examination of the flow field is currently being obtained using $100 \times 64 \times 64$ points on a CRAY-XMP with SSD memory. Computer graphics are presently being applied to study the topology of three-dimensional flow separation and the physics of the flow field in general.

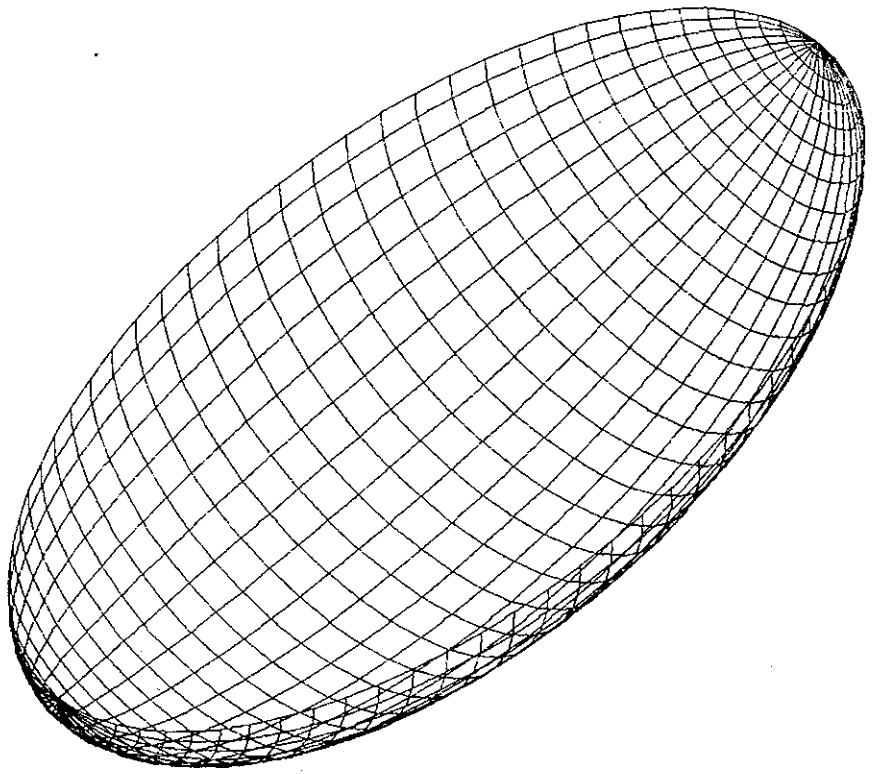


Figure 19. Surface Grid Distribution for 6-1 Ellipsoid.

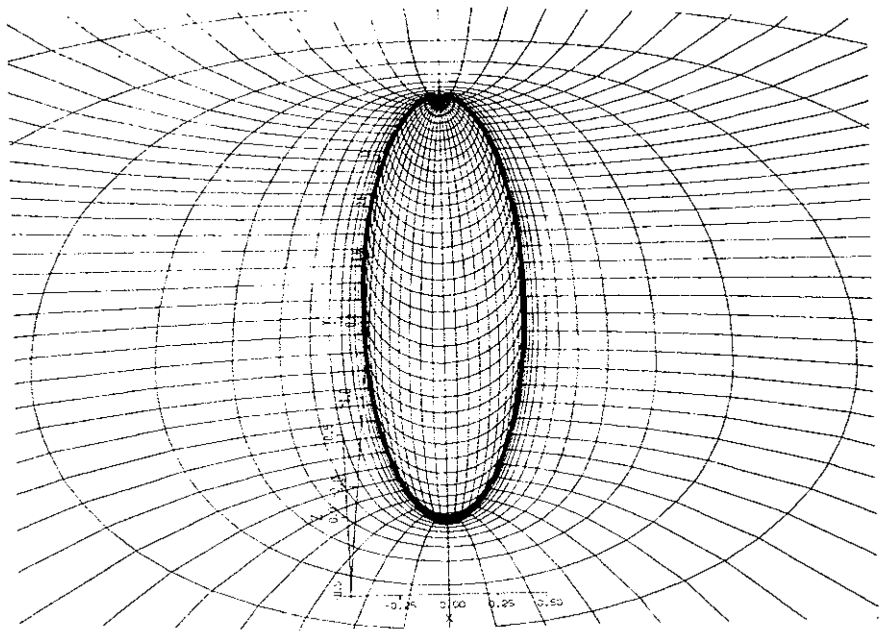


Figure 20. Grid Distribution for 6-1 Ellipsoid.

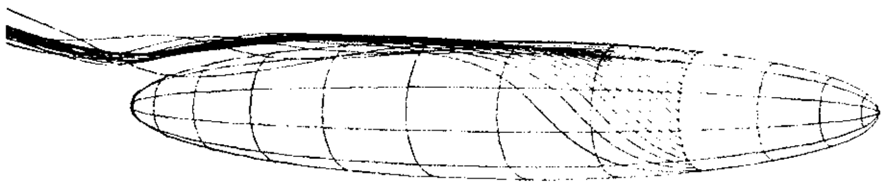


Figure 21. Particle Paths Showing Crossflow Separation.

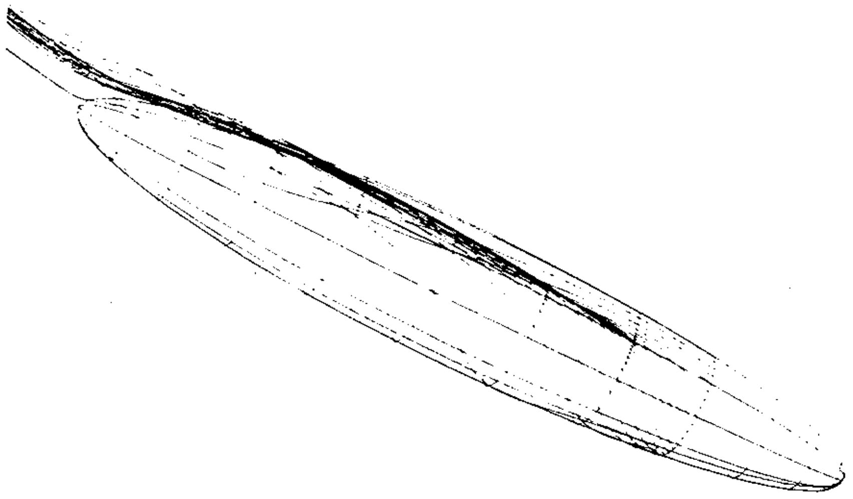


Figure 22. Particle Paths Showing Crossflow Separation.

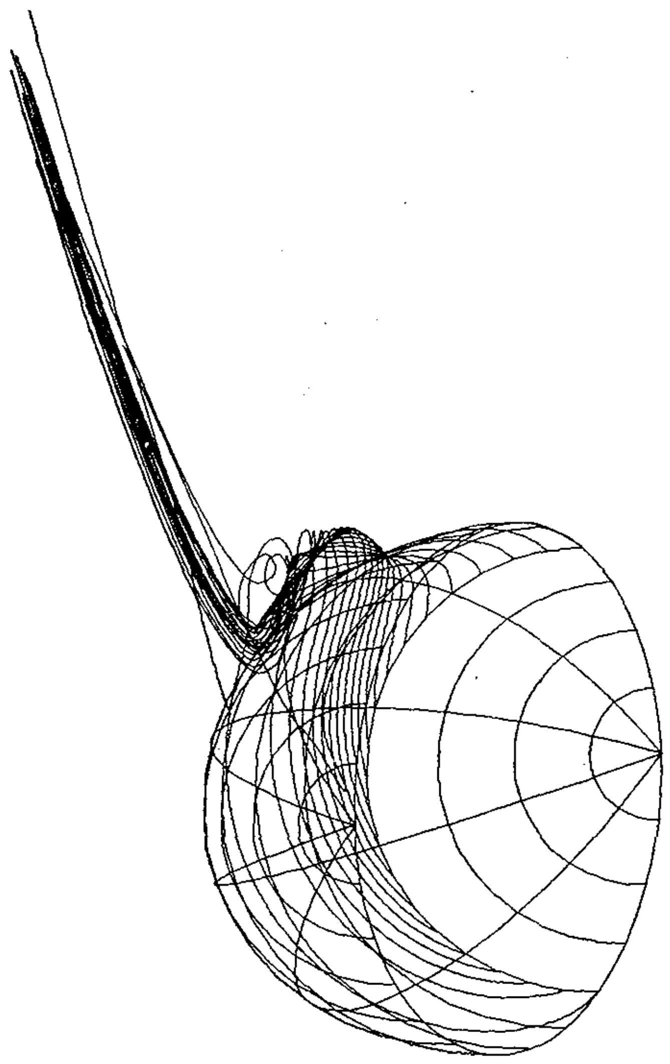


Figure 23. Particle Paths Showing Crossflow Separation.

VI) CLOSING

General purpose centrally space differenced implicit finite difference codes, ARC2D and ARC3D, developed at NASA Ames run either in inviscid or viscous mode for steady or unsteady flow. They use general coordinate systems and can be run on any smoothly varying curvilinear meshes. The codes take advantage of vectorized computer processors and have been implemented for the Control Data 205 and the CRAY 1-S and X-MP.

By using a series of established and simple procedures, a set of straightforward general purpose computer codes have been developed which are competitive with specialized codes. Changes in boundary conditions and numerical dissipation models have improved the accuracy of these codes over previous versions. Convergence characteristics, stability and robustness are greatly improved resulting in a set of excellent computational tools for application to Euler and Navier-Stokes calculations.

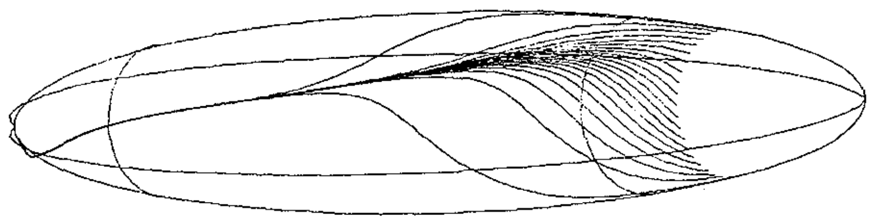


Figure 24. Simulated Oilflow On Surface Showing Separation Line

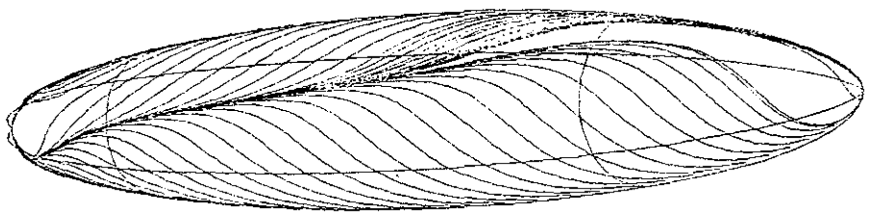


Figure 25. Simulated Oilflow On Surface Showing Separation Line

REFERENCES

1. Steger, J. L., *Implicit Finite Difference Simulation of Flow About Arbitrary Geometries with Application to Airfoils*, AIAA J., Vol. 16, 77-665 (July, 1978), p. 679.
2. Pulliam, T. H. and Steger, J. L., *Implicit Finite-Difference Simulations of Three-Dimensional Compressible Flow*, AIAA J **18** (1980), p. 159.
3. Beam, R. and Warming, R. F., *An Implicit Finite-Difference Algorithm for Hyperbolic Systems in Conservation Law Form*, J. Comp. Phys. **22** (1976,), 87-110.
4. Briley, W. R. and McDonald, H., *Solution of the Multidimensional Compressible Navier-Stokes Equations by a Generalized Implicit Method*, J. of Comp. Phy. 24 (1977), 372-397.
5. Reddy, K. C., *Pseudospectral Approximation in a Three-Dimensional Navier-Stokes Code*, AIAA J. **21, no. 8** (Aug. 1983).
6. Chow, L. J., Pulliam, T. H. and Steger, J. L., *A General Perturbation Approach for the Equations of Fluid Dynamics*, AIAA paper no. 83-1903 (1983).
7. Shang, J. S. and Hankey, W. L.,, *Numerical Solution of the Compressible Navier-Stokes Equations*

for a Three-Dimensional Corner, AIAA paper 77-169 (1977).

8. McDonald, H. and Briley, W. R., *Computational Fluid Dynamic Aspects of Internal Flows*, AIAA paper 79-1445, Proceedings of AIAA Computational Fluid Dynamics Conference, Williamsburg, Va. (1979).
9. Srinivasan, G. R., Chyu, W. J., and Steger, J. L., *Computation of Simple Three-Dimensional Wing - Vortex Interaction in Transonic Flow*, AIAA paper 81-1206 (1981).
10. Coakley, T. J., *Numerical Method for Gas Dynamics Combining Characteristic and Conservation Concepts*, AIAA Paper 81-1257 (1981).
11. Jameson, A., *Solution of the Euler Equations for Two-Dimensional Transonic Flow by a Multigrid Method*, Appl. Math. and Computation **13** (1983), 327–355.
12. Pulliam, T. H. and Chaussee, D. S., *A Diagonal Form of an Implicit Approximate Factorization Algorithm*, J C P **39** (1981), p. 347.
13. Barth, T. J. and Steger, J. L., *A Fast Efficient Implicit Scheme for the Euler and Navier-Stokes Equations Using Matrix Reduction Techniques*, Submitted to AIAA 23rd Aerospace Sciences Meeting, Reno, Nev. 1985.
14. Warming, R. F., Beam, R., and Hyett, B. J., *Diagonalization and Simultaneous Symmetrization of the Gas-Dynamic Matrices*, Math Comp **29** (1975), p. 1037.
15. Pulliam, T. H., *Artificial Dissipation Models for the Euler Equations*, AIAA paper 85-0438, AIAA 23rd Aerospace Sciences Meeting (1985).
16. MacCormack, R. and Baldwin, B., *A Numerical Method for Solving the Navier-Stokes Equations With Application to Shock Boundary Layer Interactions*, AIAA paper 75-1 AIAA 13th Aerospace Sciences Meeting, Pasadena, Calif. (1975).
17. Osher S. and Chakravarthy, S., *Upwind Schemes and Boundary Conditions with Applications to Euler Equations in General Geometries,*, J. Comp. Phys. **50** (1983), 447–481.
18. Harten, A., *A High Resolution Scheme for the Computation of Weak Solutions of Hyperbolic Conservation Laws*, J. Comp. Phys. **49** (1983), 357-393.
19. Jameson, A., Schmidt, W. and Turkel, E., *Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes*, AIAA paper 81-1259 AIAA 14th Fluid and Plasma Dynamics Conference, Palo Alto (1981).
20. Salas, M. D., Jameson, A., and Melnik, R. E., *A Comparative Study of the Nonuniqueness Problem of the Potential Equation*, AIAA paper 83-1888, AIAA 6th Computational Fluid Dynamics Conference (1983).
21. *The 1980-81 AFOSR-HTTM-Stanford Conference on Complex Turbulent Flows: Comparison of Computation and Experiment, Vol. 2, Taxonomies, Reporters' Summaries, Evaluation and Conclusions*, Eds. S. J. Kline, B. J. Cantwell and G. M. Lilley, Thermoscience Division, Stanford University, California.
22. Baldwin, B. S. and Lomax, H., *Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows,*, AIAA Paper No. 78-257 (1978).
23. Cook, P. H., McDonald, M. A. and Firmin, M. C. P., *Aerofoil RAE 2822 – Pressure Distributions, and Boundary layer and Wake Measurements*, AGARD-AR-138 (1979).
24. Mehta, U., *Reynolds Averaged Navier-Stokes Computations of Transonic Flows Around Airfoils, Presented at Second Symposium on Numerical and Physical Aspects of Aerodynamic Flows, Long Beach, Calif,*.

APPENDIX

The two-dimensional flux Jacobian matrices in generalized coordinates are $\widehat{A}$ or $\widehat{B}$ =

$$\begin{bmatrix} \kappa_t & \kappa_x & \kappa_y & 0 \\ -u\theta + \kappa_x\phi^2 & \kappa_t + \theta - (\gamma-2)\kappa_x u & \kappa_y u - (\gamma-1)\kappa_x v & (\gamma-1)\kappa_x \\ -v\theta + \kappa_y\phi^2 & \kappa_x v - (\gamma-1)\kappa_y u & \kappa_t + \theta - (\gamma-2)\kappa_y v & (\gamma-1)\kappa_y \\ \theta[\phi^2 - a_1] & \kappa_x a_1 - (\gamma-1)u\theta & \kappa_y a_1 - (\gamma-1)v\theta & \gamma\theta + \kappa_t \end{bmatrix} \tag{A.1}$$

with $a_1 = \gamma(e/\rho) - \phi^2$, $\theta = \kappa_x u + \kappa_y v$, $\phi^2 = \frac{1}{2}(\gamma-1)(u^2+v^2)$, and $\kappa = \xi$ or η for $\widehat{A}$ or $\widehat{B}$, respectively.

The two-dimensional viscous flux Jacobian is

$$\widehat{M} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ m_{21} & \alpha_1\partial_\eta(\rho^{-1}) & \alpha_2\partial_\eta(\rho^{-1}) & 0 \\ m_{31} & \alpha_2\partial_\eta(\rho^{-1}) & \alpha_3\partial_\eta(\rho^{-1}) & 0 \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} J \tag{A.2}$$

where

$$\begin{aligned} m_{21} &= -\alpha_1\partial_\eta(u/\rho) - \alpha_2\partial_\eta(v/\rho) \\ m_{31} &= -\alpha_2\partial_\eta(u/\rho) - \alpha_3\partial_\eta(v/\rho) \\ m_{41} &= \alpha_4\partial_\eta\left[-(e/\rho^2) + (u^2+v^2)/\rho\right] \\ &\quad - \alpha_1\partial_\eta(u^2/\rho) - 2\alpha_2\partial_\eta(uv/\rho) \\ &\quad - \alpha_3\partial_\eta(v^2/\rho) \\ m_{42} &= -\alpha_4\partial_\eta(u/\rho) - m_{21} \\ m_{43} &= -\alpha_4\partial_\eta(v/\rho) - m_{31} \\ m_{44} &= \alpha_4\partial_\eta(\rho^{-1}) \\ \alpha_1 &= \mu[(4/3){\eta_x}^2 + {\eta_y}^2], \quad \alpha_2 = (\mu/3)\eta_x\eta_y \\ \alpha_3 &= \mu[{\eta_x}^2 + (4/3){\eta_y}^2], \quad \alpha_4 = \gamma\mu Pr^{-1}({\eta_x}^2 + {\eta_y}^2) \end{aligned}$$

The flux Jacobian matrices have real eigenvalues and a complete set of eigenvectors. The similarity transforms are

$$\widehat{A} = T_\xi \Lambda_\xi T_\xi^{-1} \quad \text{and} \quad \widehat{B} = T_\eta \Lambda_\eta T_\eta^{-1} \tag{A.3}$$

where

$$\Lambda_\xi = \begin{bmatrix} U & & & \\ & U & & \\ & & U + a\sqrt{\xi_x^2 + \xi_y^2} & \\ & & & U - a\sqrt{\xi_x^2 + \xi_y^2} \end{bmatrix}, \tag{A.3a}$$

$$\Lambda_\eta = \begin{bmatrix} V & & & \\ & V & & \\ & & V + a\sqrt{\eta_x^2 + \eta_y^2} & \\ & & & V - a\sqrt{\eta_x^2 + \eta_y^2} \end{bmatrix}, \tag{A.3b}$$

with

$$T_\kappa = \begin{bmatrix} 1 & 0 & \alpha & \alpha \\ u & \widetilde{\kappa}_y\rho & \alpha(u+\widetilde{\kappa}_x a) & \alpha(u-\widetilde{\kappa}_x a) \\ v & -\widetilde{\kappa}_x\rho & \alpha(v+\widetilde{\kappa}_y a) & \alpha(v-\widetilde{\kappa}_y a) \\ \frac{\phi^2}{(\gamma-1)} & \rho(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v) & \alpha\left[\frac{\phi^2+a^2}{(\gamma-1)} + a\widetilde{\theta}\right] & \alpha\left[\frac{\phi^2+a^2}{(\gamma-1)} - a\widetilde{\theta}\right] \end{bmatrix} \tag{A.3c}$$

$$T_\kappa^{-1} = \begin{bmatrix} (1-\phi^2/a^2) & (\gamma-1)u/a^2 & (\gamma-1)v/a^2 & -(\gamma-1)/a^2 \\ -(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)/\rho & \widetilde{\kappa}_y/\rho & -\widetilde{\kappa}_x/\rho & 0 \\ \beta(\phi^2 - a\widetilde{\theta}) & \beta[\widetilde{\kappa}_x a - (\gamma-1)u] & \beta[\widetilde{\kappa}_y a - (\gamma-1)v] & \beta(\gamma-1) \\ \beta(\phi^2 + a\widetilde{\theta}) & -\beta[\widetilde{\kappa}_x a + (\gamma-1)u] & -\beta[\widetilde{\kappa}_y a + (\gamma-1)v] & \beta(\gamma-1) \end{bmatrix} \tag{A.4}$$

and $\alpha = \rho/(\sqrt{2}a)$, $\beta = 1/(\sqrt{2}\,\rho a)$, $\widetilde{\theta} = \widetilde{\kappa}_x u + \widetilde{\kappa}_y v$, and, for example, $\widetilde{\kappa}_x = \kappa_x/\sqrt{\kappa_x^2+\kappa_y^2}$.

Relations exist between T_ξ and T_η of the form

$$\widehat{N} = T_\xi^{-1}T_\eta, \quad \widehat{N}^{-1} = T_\eta^{-1}T_\xi \tag{A.5}$$

where

$$\widehat{N} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & -\mu m_2 & \mu m_2 \\ 0 & \mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & -\mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix} \tag{A.6a}$$

and

$$\widehat{N}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & m_1 & \mu m_2 & -\mu m_2 \\ 0 & -\mu m_2 & \mu^2(1+m_1) & \mu^2(1-m_1) \\ 0 & \mu m_2 & \mu^2(1-m_1) & \mu^2(1+m_1) \end{bmatrix} \tag{A.6b}$$

with $m_1 = \left(\widetilde{\xi}_x\,\widetilde{\eta}_x + \widetilde{\xi}_y\widetilde{\eta}_y\right)$, $m_2 = \left(\widetilde{\xi}_x\,\widetilde{\eta}_y - \widetilde{\xi}_y\widetilde{\eta}_x\right)$ and $\mu = 1/\sqrt{2}$.

It is interesting to note that the matrix $\widehat{N}$ is only a function of the metrics and not the flow variables.

In three-dimensions the Jacobian matrices $\widehat{A}$ or $\widehat{B}$ or $\widehat{C}$ =

$$\left[\begin{array}{ccccc}
\kappa_t & \kappa_x & \kappa_y & \kappa_z & 0 \\
\kappa_x\phi^2 - u\theta & \kappa_t + \theta - \kappa_x(\gamma-2)u & \kappa_y u - \kappa_x(\gamma-1)v & \kappa_z u - \kappa_x(\gamma-1)w & \kappa_x(\gamma-1) \\
\kappa_y\phi^2 - v\theta & \kappa_x v - \kappa_y(\gamma-1)u & \kappa_t + \theta - \kappa_y(\gamma-2)v & \kappa_z v - \kappa_y(\gamma-1)w & \kappa_y(\gamma-1) \\
\kappa_z\phi^2 - w\theta & \kappa_x w - \kappa_z(\gamma-1)u & \kappa_y w - \kappa_z(\gamma-1)v & \kappa_t + \theta - \kappa_z(\gamma-2)w & \kappa_z(\gamma-1) \\
-\theta\left(\gamma e/\rho - 2\phi^2\right) & \kappa_x\left(\gamma e/\rho - \phi^2\right) - (\gamma-1)u\theta & \kappa_y\left(\gamma e\rho^{-1} - \phi^2\right) - (\gamma-1)v\theta & \kappa_z\left(\gamma e\rho^{-1} - \phi^2\right) - (\gamma-1)w\theta & \kappa_t + \gamma\theta
\end{array}\right] \tag{A.7}$$

where

$$\theta = \kappa_x u + \kappa_y v + \kappa_z w$$

$$\phi^2 = (\gamma-1)\left(\frac{u^2+v^2+w^2}{2}\right)$$

with $\kappa = \xi,$ or η or ς for $\widehat{A}, \widehat{B},$ or , $\widehat{C}$ respectively.

The viscous flux Jacobian is

$$\widehat{M} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
m_{21} & \alpha_1\partial_\varsigma(\rho^{-1}) & \alpha_2\partial_\varsigma(\rho^{-1}) & \alpha_3\partial_\varsigma(\rho^{-1}) & 0 \\
m_{31} & \alpha_2\partial_\varsigma(\rho^{-1}) & \alpha_4\partial_\varsigma(\rho^{-1}) & \alpha_5\partial_\varsigma(\rho^{-1}) & 0 \\
m_{41} & \alpha_3\partial_\varsigma(\rho^{-1}) & \alpha_5\partial_\varsigma(\rho^{-1}) & \alpha_6\partial_\varsigma(\rho^{-1}) & 0 \\
m_{51} & m_{52} & m_{53} & m_{54} & \alpha_0\partial_\varsigma(\rho^{-1})
\end{bmatrix} J \tag{A.8a}$$

where

$$\begin{aligned}
m_{21} =& -\alpha_1\partial_\varsigma(u/\rho) - \alpha_2\partial_\varsigma(v/\rho) - \alpha_3\partial_\varsigma(w/\rho) \\
m_{31} =& -\alpha_2\partial_\varsigma(u/\rho) - \alpha_4\partial_\varsigma(v/\rho) - \alpha_5\partial_\varsigma(w/\rho) \\
m_{41} =& -\alpha_3\partial_\varsigma(u/\rho) - \alpha_5\partial_\varsigma(v/\rho) - \alpha_6\partial_\varsigma(w/\rho) \\
m_{51} =& \alpha_0\partial_\varsigma\left[-(e/\rho^2) + (u^2+v^2+w^2)/\rho\right] \\
& -\alpha_1\partial_\varsigma(u^2/\rho) - \alpha_4\partial_\varsigma(v^2/\rho) - \alpha_6\partial_\varsigma(w^2/\rho) \\
& -2\alpha_2\partial_\varsigma(uv/\rho) - 2\alpha_3\partial_\varsigma(uw/\rho) - 2\alpha_5\partial_\varsigma(vw/\rho) \\
m_{52} =& -\alpha_0\partial_\varsigma(u/\rho) - m_{21} \qquad m_{53} = -\alpha_0\partial_\varsigma(v/\rho) - m_{31} \\
m_{54} =& -\alpha_0\partial_\varsigma(w/\rho) - m_{41} \qquad m_{44} = \alpha_4\partial_\eta(\rho^{-1}) \\
\alpha_0 =& \gamma\mu Pr^{-1}({\varsigma_x}^2 + {\varsigma_y}^2 + {\varsigma_z}^2) \quad \alpha_1 = \mu[(4/3){\varsigma_x}^2 + {\varsigma_y}^2 + {\varsigma_z}^2], \\
\alpha_2 =& (\mu/3)\varsigma_x\varsigma_y, \quad \alpha_3 = (\mu/3)\varsigma_x\varsigma_z, \quad \alpha_4 = \mu[{\varsigma_x}^2 + (4/3){\varsigma_y}^2 + {\varsigma_z}^2], \\
\alpha_5 =& (\mu/3)\varsigma_x\varsigma_z, \qquad \alpha_6 = \mu[{\varsigma_x}^2 + {\varsigma_y}^2 + (4/3){\varsigma_z}^2],
\end{aligned} \tag{A.8b}$$

The eigensystem decomposition of the three-dimensional Jacobians have the form $\widehat{A} = T_\xi\Lambda_\xi T_\xi^{-1}$,

$\widehat{B} = T_\eta \Lambda_\eta T_\eta^{-1}$, and $\widehat{C} = T_\varsigma \Lambda_\varsigma T_\varsigma^{-1}$. The eigenvalues are

$$\begin{aligned} \lambda_1 = \lambda_2 = \lambda_3 &= \kappa_t + \kappa_x u + \kappa_y v + \kappa_z w \\ \lambda_4 = \lambda_1 + \kappa a \quad & \lambda_5 = \lambda_1 - \kappa a \\ \kappa &= \sqrt{\kappa_x^2 + \kappa_y^2 + \kappa_z^2} \end{aligned} \tag{A.9}$$

The matrix T_κ, representing the left eigenvectors, is

$$T_\kappa = \begin{bmatrix} \widetilde{\kappa}_x & \widetilde{\kappa}_y & \widetilde{\kappa}_z & \alpha & \alpha \\ \widetilde{\kappa}_x u & \widetilde{\kappa}_y u - \widetilde{\kappa}_z \rho & \widetilde{\kappa}_z u + \widetilde{\kappa}_y \rho & \alpha(u + \widetilde{\kappa}_x a) & \alpha(u - \widetilde{\kappa}_x a) \\ \widetilde{\kappa}_x v + \widetilde{\kappa}_z \rho & \widetilde{\kappa}_y v & \widetilde{\kappa}_z v - \widetilde{\kappa}_x \rho & \alpha(v + \widetilde{\kappa}_y a) & \alpha(v - \widetilde{\kappa}_y a) \\ \widetilde{\kappa}_x w - \widetilde{\kappa}_y \rho & \widetilde{\kappa}_y w + \widetilde{\kappa}_x \rho & \widetilde{\kappa}_z w & \alpha(w + \widetilde{\kappa}_z a) & \alpha(w - \widetilde{\kappa}_z a) \\ [\widetilde{\kappa}_x \phi^2/(\gamma - 1) + \rho(\widetilde{\kappa}_z v - \widetilde{\kappa}_y w)] & [\widetilde{\kappa}_y \phi^2/(\gamma - 1) + \rho(\widetilde{\kappa}_x w - \widetilde{\kappa}_z u)] & [\widetilde{\kappa}_z \phi^2(\gamma - 1) + \rho(\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)] & \alpha\left[(\phi^2 + a^2)/(\gamma - 1) + \widetilde{\theta} a\right] & \alpha\left[(\phi^2 + a^2)/(\gamma - 1) - \widetilde{\theta} a\right] \end{bmatrix} \tag{A.10}$$

where

$$\alpha = \frac{\rho}{\sqrt{2}a}, \quad \widetilde{\kappa}_x = \frac{\kappa_x}{\kappa}, \quad \widetilde{\kappa}_y = \frac{\kappa_y}{\kappa}, \quad \widetilde{\kappa}_z = \frac{\kappa_z}{\kappa}, \quad \widetilde{\theta} = \frac{\theta}{\kappa}$$

The corresponding T_κ^{-1} is

$$T_\kappa^{-1} = \begin{bmatrix} \widetilde{\kappa}_x \left(1 - \phi^2/a^2\right) - (\widetilde{\kappa}_z v - \widetilde{\kappa}_y w)/\rho & \widetilde{\kappa}_x(\gamma - 1)u/a^2 & \widetilde{\kappa}_x(\gamma - 1)v/a^2 + \widetilde{\kappa}_z/\rho & \widetilde{\kappa}_x(\gamma - 1)w/a^2 - \widetilde{\kappa}_y/\rho & -\widetilde{\kappa}_x(\gamma - 1)/a^2 \\ \widetilde{\kappa}_y \left(1 - \phi^2/a^2\right) - (\widetilde{\kappa}_x w - \widetilde{\kappa}_z u)/\rho & \widetilde{\kappa}_y(\gamma - 1)u/a^2 - \widetilde{\kappa}_z/\rho & \widetilde{\kappa}_y(\gamma - 1)v/a^2 & \widetilde{\kappa}_y(\gamma - 1)w/a^2 - \widetilde{\kappa}_x/\rho & -\widetilde{\kappa}_y(\gamma - 1)/a^2 \\ \widetilde{\kappa}_z \left(1 - \phi^2/a^2\right) - (\widetilde{\kappa}_y u - \widetilde{\kappa}_x v)/\rho & \widetilde{\kappa}_z(\gamma - 1)u/a^2 + \widetilde{\kappa}_y/\rho & \widetilde{\kappa}_z(\gamma - 1)v/a^2 - \widetilde{\kappa}_x/\rho & \widetilde{\kappa}_z(\gamma - 1)w/a^2 & -\widetilde{\kappa}_z(\gamma - 1)/a^2 \\ \beta(\phi^2 - \widetilde{\theta} a) & -\beta[(\gamma - 1)u - \widetilde{\kappa}_x a] & -\beta[(\gamma - 1)v - \widetilde{\kappa}_y a] & -\beta[(\gamma - 1)w - \widetilde{\kappa}_z a] & \beta(\gamma - 1) \\ \beta(\phi^2 + \widetilde{\theta} a) & -\beta[(\gamma - 1)u + \widetilde{\kappa}_x a] & -\beta[(\gamma - 1)v + \widetilde{\kappa}_y a] & -\beta[(\gamma - 1)w + \widetilde{\kappa}_z a] & \beta(\gamma - 1) \end{bmatrix} \tag{A.11}$$

where

$$\beta = \frac{1}{\sqrt{2}\rho a}$$